UNIVERSITÁ DI PISA
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE

ENHANCING CONTENT-BASED IMAGE RETRIEVAL USING AGGREGATION OF BINARY FEATURES, DEEP LEARNING, AND SUPERMETRIC SEARCH

DOCTORAL THESIS

Author
Lucia Vadicamo

Tutor (s)
Dr. Giuseppe Amato, Dr. Fabrizio Falchi, Prof. Francesco Marcelloni

Reviewer (s)
Prof. Stéphane Marchand-Maillet, Prof. Pavel Zezula

The Coordinator of the PhD Program
Prof. Marco Luise

Pisa, May 2018

XXX

To my family

"*The mere formulation of a problem is far more essential than its solution, which may be merely a matter of mathematical or experimental skill. To raise new questions, new possibilities, to regard old problems from a new angle requires creative imagination and marks real advances in science.*"
Albert Einstein

Acknowledgements

This thesis would not have been possible without the support of many people.

First of all, I would like to thank my supervisors Dr. Giuseppe Amato and Dr. Fabrizio Falchi for teaching me what it means to do research and for their professional guidance, continuous encouragement, inspiring ideas and friendly mentorship over the course of my PhD as well as my work at the CNR.

I am highly grateful to Dr. Claudio Gennaro who has collaborated with me in several research activities, and to Prof. Richard Connor who has involved me in the fascinating research on supermetric spaces. They both provided me with many valuable insights. The last two chapters of this thesis would not have been written without the collaborations I had with them.

Many thanks go also to Dr. Fausto Rabitti for his valuable advice and for providing me with many opportunities for professional growth, such as funding for attending international conferences.

Moreover, I would like to acknowledge Prof. Francesco Marcelloni for supervising my PhD, and Prof. Pavel Zezula and Prof. Stéphane Marchand-Maillet for their time and dedication in reviewing my thesis.

Special thanks go to all the past and present colleagues at the Networked Multimedia Information Systems laboratory, especially Paolo Bolettieri, Franco Alberto Cardillo, Fabio Carrara, Franca Debole, Alejandro Moreo, Alessandro Nardi, Claudio Vairo, and Loredana Versienti for their encouragement, stimulating collaborations, interesting conversations, amusing coffee breaks and "crazy deals"!

I am very grateful to have so many amazing friends, including Giulia, Michele, Elisa, Mario, Enrico, Paola, Maria Laura, Daniela, Tiziano, Daniela, Roberto, Simona, Carlo, Sergio, Flavia, Federico, and my Lunadonda-friends. I would like to thank them for all the fun and good times we shared together (and for having endured all the tedious talks about deadlines/thesis/exams/paper submissions!). A special mention goes to Laura, my dearest friend, for having been very close to me in these years.

My last but not least thanks are for my family. I would like to thank my parents and my brother for supporting me in every step of my life, loving me, believing in me and stimulating me to do better and better. Many thanks go to my sister-in-law, Maria, for the same reasons and for giving me the best gift of ever: my lovely niece Arianna. Finally, my deepest thanks go to my husband, Marco, for continuously encouraging me and keeping me going when times were tough. He probably hates this thesis due all the evenings I spent writing it instead of spending time together, but he deserves credits for every single page.

Summary

THE millions of images shared every day on social media is just a tip of the iceberg of the current phenomenon of visual data explosion, which places a great demand on scalable Content-Based Image Retrieval (CBIR) systems. CBIR allows organizing and searching image collections on the basis of image visual contents, that is without using text or other metadata. The problem of content-based search is addressed in this thesis by investigating and proposing efficient and effective methods that support three fundamental stages of a CBIR system, namely the numerical representation of the image visual content (feature extraction), the processing/indexing of the image features, and the query-by-example search.

Concerning the image representation we investigate and experimentally compare Convolutional Neural Network (CNN) features, methods for aggregating local features, and their combination. We show that very high effectiveness is achieved combining CNN features and aggregation methods; moreover, in order to improve the efficiency we investigate the use of the aggregation methods on the top of binary local features. In particular, we propose the *BMM-FV* which allows encoding a set of binary vectors into a single descriptor. An extensive experimental evaluation on benchmark datasets shows that our BMM-FV outperforms other methods for aggregating binary local features and achieves high retrieval performance when combined with the CNN features.

Secondly, we propose an efficient and effective technique, called *Deep Permutation*, to index deep features (such as CNN features) using a permutation-based approach. Moreover, we propose the *Block-wise Surrogate Text Representation* to represent and index compound metric objects, including the VLAD image descriptors, using off-the-shelf text search engine.

Finally, we address the image search task in the general context of similarity search in metric space, which is a framework suitable for a large number of applications and data types. Most metric indexing and searching mechanisms rely on the triangle inequality, which allows deriving bounds on the distance between data objects. The distance bounds are used to efficiently exclude partition of the data that do not contain solutions to a given query. We reread foundations of metric search from a geometrical point of view starting from the observation that the triangle inequality is equivalent to a discrete geometric condition defined in term of finite isometric embeddings into Euclidean spaces. We show that there exists a large class of metric spaces, the *supermetric* ones, meeting the four-point property that is a property stronger than the triangle inequality. Moreover, we show that many supermetric spaces commonly used in applications have a further property called n-point property. The main outcome of our study is showing how these geometric properties can be used to improve the similarity search in supermetric spaces by 1) deriving distance bounds that are tighter than that relied on the triangle inequality and, thus, allowing better space pruning; 2) defining novel partitioning and indexing mechanisms; 3) proposing a promising approach to embed a supermetric space into a finite-dimensional Euclidean space, which turns out to have implications not only in the similarity search context but also in other applicative tasks such, as the dimensionality reduction. We prove the validity of our approaches both theoretically and experimentally.

Sommario

La dilagante diffusione di fotocamere digitali, come quelle integrate in cellulari e tablet, e la disponibilità di numerosi sistemi di archiviazione tramite Internet favoriscono oggigiorno una massiccia produzione di contenuti multimediali, ponendo al contempo il problema della gestione di grandi archivi visuali. Inoltre, il trend seguito dagli utenti dei social media, abituati a scattare e condividere le proprie foto senza descriverne propriamente il contenuto, rispecchia la diffusa mancanza di metadati associati alle immagini. Tali fattori hanno reso opportuno lo studio e lo sviluppo di sistemi di *Content Based Image Retrieval* (CBIR), ossia sistemi capaci di archiviare e reperire le immagini utilizzandone il loro "contenuto visivo". In questa tesi, il problema della ricerca per similarità visuale viene affrontato analizzando e proponendo algoritmi efficaci ed efficienti a supporto di tre fasi fondamentali di un qualsiasi sistema CBIR, ossia: 1) la scelta di un'opportuna rappresentazione matematica del contenuto delle immagini (estrazione di *feature*); 2) l'elaborazione e l'indicizzazione di tali rappresentazioni; 3) la fase di ricerca data un'immagine come *query*.

Come primo contributo, la tesi presenta un'estensiva analisi di varie tecniche per la rappresentazione delle immagini, quali le *Convolutional Neural Network* (CNN) *feature* e le tecniche di aggregazione di feature locali di immagini (come BoW, VLAD e FV). I risultati sperimentali, mostrano che è possibile ottenere un'elevata efficacia combinando le CNN *feature* con le tecniche di aggregazione. Al fine di raggiungere migliori performance in termini di efficienza, è stato investigato anche l'uso di tecniche di aggregazione di *feature binarie*, che rispetto alle *feature* non-binarie hanno una minore occupazione di memoria e sono fino a due ordini di grandezza più veloci da calcolare. In particolare, è stata proposta la tecnica BMM-FV che consente si aggregare un insieme di vettori binari in un unico descrittore. Un'approfondita analisi sperimentale effettuata su dataset di *benchmark* ha mostrato come le performance dei BMM-FV siano migliori rispetto a quelle di altre aggregazioni di *feature* locali binarie precedentemente proposte in letteratura; inoltre, la combinazione di CNN *feature* con il BMM-FV ha ottenuto un'elevata efficacia, comparabile a quella raggiunta combinato le CNN *feature* con le più costose aggregazioni di *feature* non-binarie.

La seconda parte della tesi è dedicata a tecniche di elaborazione ed indicizzazione dei descrittori estratti dalle immagini. In questo contesto, è stato proposto un metodo per l'indicizzazione di descrittori "a blocchi" (come ad esempio il VLAD) che, utilizzando tecniche legate al *Permutation-Based Indexing*, permette di trasformare il descrittore iniziale in una "codifica testuale" detta *Blockwise Surrogate Text Representation* (BSTR). Il vantaggio della codifica BSTR è quello di rendere scalabile la ricerca per similarità visuale attraverso metodologie di indicizzazione e di ricerca tipicamente utilizzati per il testo (come l'uso delle liste invertite), dando quindi la possibilità di utilizzare librerie e software largamente diffusi (come *Apache Lucene*). Inoltre, è stata proposta una tecnica, detta *Deep Peurmuations*, per la rappresentazione e l'indicizzazione delle emergenti *deep feature*. La tecnica proposta permette di associare a ciascuna *deep feature* una permutazione da indicizzazione successivamente tramite tecniche

basate sulle permutazioni. L'approccio proposto ha mostrato vantaggi sia in termini di efficienza che di efficacia.

Infine, la tesi affronta il problema della ricerca per similarità visuale nel contesto generale della ricerca per similarità in spazi metrici. La maggior parte dei meccanismi di indicizzazione e ricerca metrica definiti in letteratura utilizzano la disuguaglianza triangolare per derivare limiti (*bound*) sulla distanza tra la *query* e gli oggetti del dataset al fine di ridurre il numero di distanze calcolate nella fase di ricerca. Infatti, i *bound* sulla distanza possono essere usati per includere o escludere alcune partizioni del dataset che, rispettivamente, contengono o non contengono soluzioni per una determinata *query* (*space pruning*). Lo studio presentato in questa tesi mostra come alcuni fondamenti della teoria sulla ricerca in spazi metrici possano essere riletti in termini geometrici, utilizzando immersioni isometriche finite in spazi Euclidei. In particolare è stata analizzata la classe degli spazi supermetrici che hanno una proprietà geometrica più forte della disuguaglianza triangolare, detta proprietà dei quatto punti. È stato inoltre dimostrato che molti degli spazi supermetrici comunemente usati in letteratura o nelle applicazioni, soddisfano anche la proprietà degli n punti, definita in termini di immersioni isometriche in spazi Euclidei n-dimensionali. Il risultato principale dello studio presentato è stato dimostrare come queste proprietà possano essere utilizzate per migliorare la ricerca per similarità negli spazi supermetrici. Tale miglioramento è stato ottenuto 1) derivando *bound* sulle distanze che sono più restrittivi di quelli ottenuti usando la disuguaglianza triangolare, e che quindi permettendo un migliore *pruning* dello spazio di ricerca; 2) definendo nuovi meccanismi di partizionamento e di indicizzazione; 3) proponendo un innovativo approccio per l'immersione di uno spazio supermetrico in uno spazio Euclideo finito-dimensionale, il quale ha mostrato di avere importanti risvolti non solo nel contesto della ricerca per similarità, ma anche in altri contesti applicativi quali la riduzione di dimensionalità dei dati. Le tecniche proposte sono state analizzate e validate sia teoricamente che sperimentalmente.

List of publications

International Journals

1. Connor, R., Vadicamo, L., Cardillo, F. A., Rabitti, F. (2018) Supermetric Search. *Information Systems*. In press.

2. Amato, G., Falchi, F., and Vadicamo, L. (2017). Aggregating binary local descriptors for image retrieval. *Multimedia Tools and Applications (MTAP)*. pp. 1-31.

3. Amato, G., Falchi, F., and Vadicamo, L. (2016). Visual recognition of ancient inscriptions using Convolutional Neural Network and Fisher Vector. *Journal on Computing and Cultural Heritage (JOCCH)*, 9(4), 21.

4. Connor, R., Cardillo, F. A., Vadicamo, L., and Rabitti, F. (2016). Hilbert exclusion: improved metric search through finite isometric embeddings. *ACM Transactions on Information Systems (TOIS)*, 35(3), 17.

International Conferences/Workshops with Peer Review

1. Vadicamo, L., Carrara, F., Cimino, A., Cresci, S., Dell'Orletta, F., Falchi, F., Tesconi, M. (2017, October). Cross-Media Learning for Image Sentiment Analysis in the Wild. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, (pp. 308-317).

2. Connor, R., Vadicamo, L., Rabitti, F. (2017, October). High-Dimensional Simplexes for Supermetric Search. In *Proceedings of the International Conference on Similarity Search and Applications (SISAP 2017)*, (pp. 96-109). Springer International Publishing.

3. Amato, G., Mannocci, A., Vadicamo, and L., Zoppi, F. (2017, January). Coping with interoperability in cultural heritage data infrastructures: the Europeana network of Ancient Greek and Latin Epigraphy. In *Proceedings of the 6th AIUCD Conference 2017 (AIUCD 2017)*, (pp. 211-215).

4. Amato, G., Falchi, F., Gennaro, C., and Vadicamo, L. (2016, October). Deep Permutations: Deep Convolutional Neural Networks and Permutation-Based Indexing. In *Proceedings of the International Conference on Similarity Search and Applications (SISAP 2016)* (pp. 93-106). Springer International Publishing.

5. Connor, R., Vadicamo, L., Cardillo, F. A., and Rabitti, F. (2016, October). Supermetric search with the four-point property. In *Proceedings of the International Conference on Similarity Search and Applications (SISAP 2016)* (pp. 51-64). Springer International Publishing.

6. Amato, G., Falchi, F., Rabitti, F., and Vadicamo, L. (2016, May). Combining Fisher Vector and Convolutional Neural Networks for Image Retrieval. In *CEUR Workshop Proceedings of the 7th Italian Information Retrieval Workshop (IIR 2016)*. (Vol. 1653).

7. Amato, G., Falchi, F., and Vadicamo, L. (2016, February). How Effective Are Aggregation Methods on Binary Features?. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, (VISIGRAPP 2016)*, (pp. 566-573).

8. Amato, G., Bolettieri, P., Falchi, F., Gennaro, C., and Vadicamo, L. (2016, February). Using Apache Lucene to Search Vector of Locally Aggregated Descriptors. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, (VISIGRAPP 2016)*, (pp. 383-392).

9. Bolettieri, P., Casarosa, V., Falchi, F., Vadicamo, L., Martineau, P., Orlandi, S., and Santucci, R. (2015, October). Searching the EAGLE Epigraphic Material Through Image Recognition via a Mobile Device. In *Proceedings of the International Conference on Similarity Search and Applications (SISAP 2015)* (pp. 351-354). Springer International Publishing.

10. Amato, G., Bolettieri, P., Falchi, F., Rabitti, F., and Vadicamo, L. (2015, May). Visual Recognition in the EAGLE Project. In *CEUR Workshop Proceedings of the 6th Italian Information Retrieval Workshop (IIR 2015)*. (Vol. 1404).

Others

1. Amato, G.; Bolettieri, P.; Falchi, F. and Vadicamo, L. (2016). Sistema di riconoscimento delle immagini e mobile app. *Forma Vrbis*. (Vol. 11, pp. 22-25). E.S.S. Editorial Service System. *(National Journal)*

List of Abbreviations

A

A-KAZE	Accelerated-KAZE. 13, 67, 71, 72, 77–79
AESA	Approximating and Eliminating Search Algorithm. 38

B

BMM	Bernoulli Mixture Model. 67, 69–73, 75
BMM-FV	Fisher Vector computed using a Bernoulli Mixture Model. 3, 4, 52, 67–80, 149, 151, 152
BoW	Bag-of-Words. 3, 10, 13–15, 52, 57, 62, 64, 67, 68, 71–76
BPP	Balancing Pivot-Position occurrences. 42, 151
BRIEF	Binary Robust Independent Elementary Features. 11–13
BST	Bisector Tree. 31, 37, 114, 115, 131
BSTR	Blockwise Surrogate Text Representation. 81, 82, 85–87, 89, 90, 150

C

CBIR	Content-Based Image Retrieval. 1–4, 6–8, 10, 14, 23, 51, 98, 149
CNN	Convolutional Neural Network. 3, 4, 19, 20, 22, 49–54, 57, 60–64, 66–68, 71, 72, 75–81, 90, 94, 145, 149–151

D

DiSAT	Distal Spatial Approximation Tree. 38, 111, 117
DSAT	Dynamic Spatial Approximation Tree. 38

F

FFT	Farthest-First Traversal. 42, 94, 95, 151
FIM	Fisher Information Matrix. 16, 69, 70, 153–155
FV	Fisher Vector. 3, 10, 14, 16, 17, 22, 51–54, 56–60, 62–64, 66–81, 149–151

G

GHT Generalized Hyperplane Tree. 31, 37, 114, 115
GMM Gaussian Mixture Model. 17, 56, 58, 59, 69–76

I
IDim Intrinsic Dimensionality. 28, 111, 112, 145

L
LAESA Linear AESA. 38, 39, 139–143
LATCH Learned Arrangements of Three patCHes codes. 12, 13, 67, 71, 72, 77–79
LMDS Landmark MDS. 138, 139
LRT Linear Regression Tree. 127, 129–131, 148
LSH Locality-Sensitive Hashing. 24, 41

M
M-index Metric Index. 32, 42, 45
mAP mean Average Precision. 9, 55, 57–64, 66, 72–74, 76–80, 87–89, 96, 97
MBT Monotonous Bisector Tree. 31, 37, 114, 115, 127, 129, 130, 141–143
MDS Multidimensional Scaling. 138
MI-File Metric Inverted File. 42, 45, 82, 90, 151

O
ORB Oriented FAST and Rotated BRIEF. 3, 10–13, 51, 52, 67, 71–80

P
PBI Permutation-Based Indexing. 24, 32, 41, 43, 45, 81, 85, 90
PCA Principal Component Analysis. 14, 15, 17, 28, 41, 56, 57, 59–64, 66–68, 72, 74–76, 78–80, 114, 138, 139
PP-Index Permutation Prefix Index. 42, 45
PPP-Index Pivot Permutation Prefix Index. 42, 45
PSIS Pivoted Space Incremental Selection. 42, 151

R
R-MAC Regional Maximum Activation of Convolutions. 22, 97, 151
RANSAC RANdom SAmple Consensus. 10, 14, 62, 64
ReLU Rectified Linear Unit transform. 20, 22, 57, 59, 60, 62, 63, 72, 77, 93, 97, 145
RNN Recurrent Neural Networks. 19, 97, 151

S
SAT Spatial Approximation Tree. 37, 38
SIFT Scale-Invariant Feature Transform. 3, 10–13, 51–53, 56–58, 62, 67, 68, 74–76, 78–80, 87, 149, 150
STR Surrogate Text Representation. 42, 46, 81, 82, 84–87, 89, 150
SURF Speeded Up Robust Features. 10–13

T

tf-idf	term frequency-inverse document frequency. 14, 62, 71, 87–90

V

VIR	Visual Information Retrieval library. 55, 56, 72, 87
VLAD	Vector of Locally Aggregation Descriptors. 3, 4, 10, 14, 15, 22, 52, 57, 62, 64, 67, 68, 71–76, 81, 82, 85–87, 89, 90, 150, 151
VPT	Vantage Point Tree. 30, 37, 131

List of Figures

List of Tables

Contents

CHAPTER 1

Introduction

People seeing this page for the first time are likely to have a look at the picture at the end of the page before starting to read this paragraph. The reason is that our brain reacts differently to visual stimuli than written words. Moreover, it processes visual contents much faster than texts. That is why images have always been one the most effective ways of communicating ideas and sharing knowledge. Cave painting, figurative art, photos, and digital images all share a common power: they are able to grab our attention easily, fast delivery a message, make it possible to transcend language barriers and communicate with people of different country and culture. Nowadays, millions of images are shared daily on social media. For example, just Instagram, which currently has 715 million active users and a total of 34.7 billion shared images, registers an average of 52 million photo uploads every day [11].

Ubiquitous digital cameras, easy Internet access, and new storage technologies have led to an explosion of digital visual data. However, most of these data are poorly annotated or not annotated at all, which places a great demand on scalable Content-Based Image Retrieval (CBIR) systems. CBIR aims to

Figure 1.1: *Human mind processes images much faster than texts*

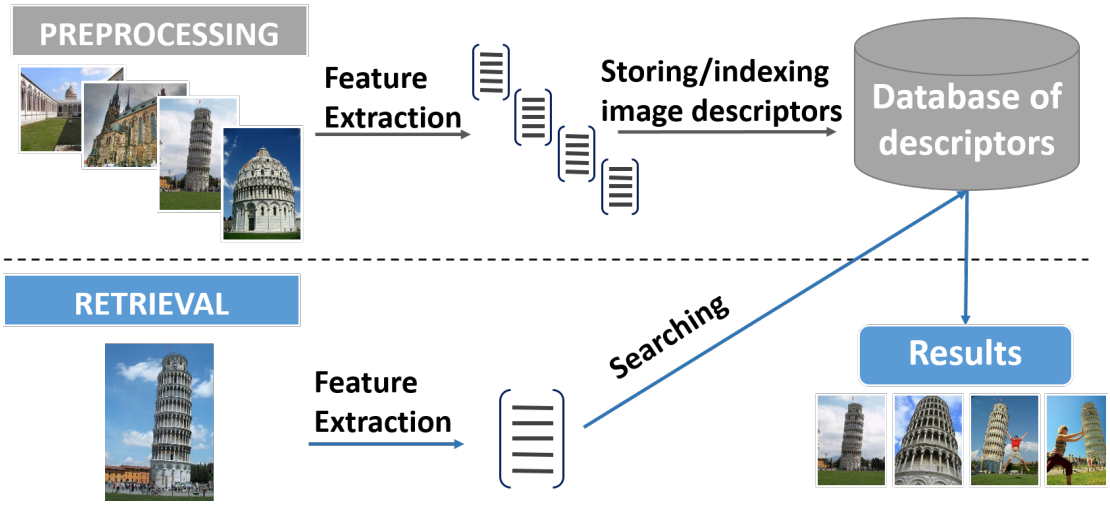


Figure 1.2: *Overview of a generic visual information retrieval system.*

organize and search digital collections of images on the basis of their visual contents, *i.e* without using text or other metadata, but analysing the images. So, they offer a promising way for searching visual archives like "using a picture as query", which perhaps is the easiest way for a user to obtain information about a visible object/instance of interest. This search paradigm meets the needs of several applications. For example, the Museum Mobile Guide (MMG) of the *Pinacoteca di Brera* [7] is a free mobile app used for searches based on pictures taken by handheld devices: a user can point his/her mobile phone camera at a painting, or any artwork of the gallery; if the MMG finds it in its database it will provide the user with the description of the photographed object, also given the possibility to listen to an audio-guide message.

The content-based search is a challenging task: as the old saying goes, a picture is worth a thousand words, but how automatically identifies some of these "words" from a simple array of pixel intensities? How to manage and compare images in order to find data with similar contents? Zhou et al. [277] in their recent survey on CBIR state that "*technically speaking, there are three key issues in content-based image retrieval: image representation, image organization, and image similarity measurement*". In this thesis, we focus on all these three key aspects, exemplified in Figure 1.2. Firstly, images need to be represented by descriptors that capture some distinctive image characteristics (*features*). Those descriptors usually lie in metric space, where it is possible to use a distance function for the image comparison, or more generally, in a space where a function asserting the (dis-)similarity of two image descriptors is defined. The image features are then managed and stored in a database so that, given an image as a query, the system searches for the database objects whose features are the most similar to the query ones. This is only the skeleton of a visual information retrieval system, which typically involves other important aspects such as visualizations, clusterings, and relevance feedbacks, to name but a few. Moreover, a further key aspect to develop retrieval systems able to operate on a large scale is using ad hoc indexing and searching algorithms, like those investigated in the last two decades in the field of *similarity search* [271].

Image retrieval on a large scale has to address at the same time two old and ever-present issues: *efficiency* and *effectiveness*, which can be described as follows.

Efficiency: Search results should be obtained quickly (ideally in real time), with low computational and storage costs.

Effectiveness: Search results should be pertinent to the submitted queries in order to satisfy the users' expectations.

Moreover, the used algorithm should be *scalable* in order to handle increasing volume of data. The performance of a CBIR system is determined by the adopted image features, indexing and searching

methods, while the trade-off between efficiency and effectiveness highly depends on the applicative scenario and the available resources. For example, a system running on a mobile device requires efficient solutions (real-time response and low memory occupation), eventually at the expense of a degradation in the quality of the results. In other cases, the effectiveness of the overall retrieval system may be the main concern.

Thesis Objectives and Contributions The research presented in this thesis has been carried out during the three-year period of the PhD Program (2014-2017). Our focus has been on investigating new ways for an efficient and effective content-based image retrieval. In particular, we have stated the following objectives.

(a) *Investigate and propose efficient and effective image descriptors*

(b) *Improve descriptor processing for efficient indexing*

(c) *Propose efficient approaches to perform image search or, more generally, similarity search*

In compliance with the objectives specified above, our main contributions are:

- *Filling a knowledge gap in evaluating the effectiveness of aggregations of binary local features [obj. (a)]:* During the last decades, various local features have been proposed and used to support CBIR and object recognition tasks. Local features, like SIFT [173], allow effectively matching local structures between images, but thousands of local features are usually extracted from each image. Therefore the cost of extracting and pairwise comparing local descriptors becomes a bottleneck when mobile devices and/or large database are used. On one hand, the cost for extracting, representing, and comparing local visual descriptors has been dramatically reduced by recently proposed binary local features, *e.g.* ORB [221]. On the other hand, state-of-the-art aggregation techniques (*e.g.* BoW [233], VLAD [143], and Fisher Vector (FV) [208]) provide effective summaries of all the extracted features of an image into a single descriptor, allowing to speed up and scale up the image search. Only a few works have recently mixed together these two research directions, defining aggregation methods for binary local features. Even so, to the best of our knowledge, those approaches have never been systematically evaluated and compared before our study: in Chapter 3 we provide a comparative analysis of aggregations of binary local features on benchmarks for image retrieval.

- *Deriving a new encoding schema to aggregate binary features, named BMM-FV [obj. (a)]:* The FV [208] is an encoding scheme that has attracted much attention thank to its effectiveness in both image classification and image search. The FV uses a "probabilistic visual vocabulary" to transform an incoming set of descriptors into a fixed-sized vector representation. The Gaussian Mixture Model is originally used to model the descriptors. In Chapter 3 and Appendix A, we mathematically derive a formulation of the FV, named BMM-FV, that uses a Bernoulli Mixture model in order to encode binary vectors. We experimentally show that the proposed BMM-FV outperforms other state-of-the-art methods for aggregating binary local features.

- *Investigating the combination of aggregations of local features and Convolutional Neural Network (CNN) features [obj. (a)]:* Recently, features extracted using deep learning approaches [161] have brought about breakthroughs in processing multimedia contents. In particular, the CNN features [217] have achieved very high effectiveness in various vision and retrieval tasks. When compared with local features, like SIFTs, the CNN features show complementary behaviour under some image transformations. This motivated researchers to explore the combination of deep learning features with other local feature-based descriptors. In Chapter 3 we perform an extensive experimental evaluation of the combinations of encodings of local features and CNN features for visual recognition and retrieval tasks.

- *Proposing novel approaches to process some common image descriptors for efficient and effective indexing [obj. (b)]:* In the realm of similarity search, many metric indexing techniques are

available, notable including permutation-based indexing. All these techniques share a fundamental aspect: they are flexible data structure able to deal with a wide spectrum of applications and data objects. The extensibility is a key element of metric indexes, however, when the domain of application is fixed (*e.g.* indexing specific image features) it is possible to pre-process the data objects before indexing them in order to obtain better performance or use an off-the-shelf search engine. In Chapter 4, we propose a *blockwise* permutation representation that allows to effectively index the Vector of Locally Aggregation Descriptors (VLAD) [143] (or other blockwise) descriptors by using conventional text search engine. In the same chapter, we also present a very efficient and effective approach for representing emerging deep CNN features as permutations, called *Deep Permutations*, to be then indexed with permutation-based methods.

- *Investigating supermetric search and proposing a novel pruning rule, called Hilbert Exclusion [obj. (c)]:* Metric search is concerned with the efficient evaluation of queries in metric space, where the distance computation is the only operation available to compare two objects. In general, a set of data is processed and partitioned in such a way that, when a further object is presented as a query, those objects closer to the query can be efficiently found. Most mechanisms use the triangle inequality of the metric governing the space to avoid unnecessary distance calculation, *e.g.* determine subsets of the data that do not need to be exhaustively checked. In Chapter 5 we examine the class of *supermetric* spaces that have a stronger property, called *four-point* property that allows isometrically embedding any 4 points of the space into a 3D Euclidean space. This property gives stronger geometric guarantees, and one in particular, which we name the *Hilbert Exclusion* property, allows any indexing mechanism which uses hyperplane partitioning to perform better. One outcome of this observation is that a number of state-of-the-art indexing mechanisms over supermetric spaces can be easily refined to give a significant increase in performance; furthermore, the improvement given is greater in higher dimensions. This, therefore, leads to a significant improvement in the cost of metric search in these spaces. Moreover, we show how the four-point property can be used to derive bounds on the actual distance and define novel partitioning approaches that are possible only on supermetric spaces. To this respect, we define a novel indexing structure (the *Linear Regression tree*) for supermetric spaces, as an example of the new area of investigation opened up by our research.

- *Proposing the n-Simplex projection [obj. (b)-(c)]:* Many common metric spaces, including the ones with Euclidean, Cosine, Jensen-Shannon, and Quadratic form distances, meet the n*-point property* that allows isometrically embedding any n points of the space into a $(n-1)$-dimensional Euclidean space. In Chapter 5 we show that for these spaces it is possible to project *all* the data objects into a finite-dimensional Euclidean space by building some simplexes whose edge lengths corresponds to the distance among a set of reference objects. This projection is particularly useful when the size of the data is large or the original distance is very expensive (like the Jensen-Shannon distance) since it allows transforming the data in a finite-dimensional Euclidean space by using the distance of each data object from a set of n reference objects. Moreover, in the projected space we derive distance upper and lower bounds that converge to the original distance as the number of dimensions n becomes higher. Thanks to this property our n*-Simplex* projection shows promising results for either metric indexing and dimensionality reduction tasks.

Outline of the Thesis

The structure of this thesis is summarized in Figure 1.3. **Chapter 2** embraces background and literature review of CBIR and similarity search in metric spaces. It also provides an introduction to Deep Learning and CNN. In **Chapter 3**, we evaluate the state-of-the-art aggregations of local features, the CNN features and their combinations for visual recognition. In particular, we perform a thoughtful comparison of these techniques for recognizing ancient inscriptions, such as Latin and Greek epigraphs, and other objects related to cultural heritage. Then, we investigate the possibility of aggregate binary local features, and we propose our BMM-FV encoding. We then perform a complete comparison among the state-of-the-art aggregation methods applied to binary local features. The mathematical derivation of the BMM-FV is reported in Appendix A. **Chapter 4** explores new solutions to process VLAD and CNN

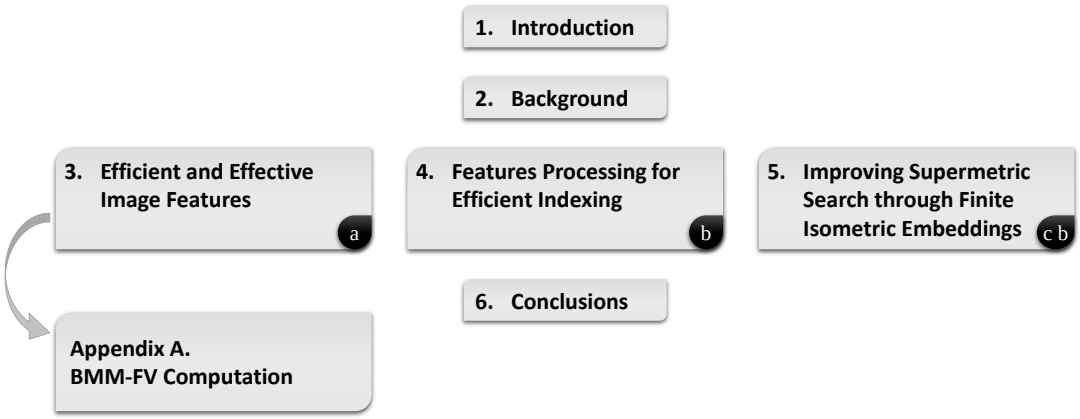


Figure 1.3: *Visualization of the thesis structure indicating where the main dissertation objectives (points (a), (b), and (c)) are addressed.*

features before indexing them with permutation-based approaches. We evaluate the performance of the proposed approach by a series of experiments. In **Chapter 5** we focus on metric search and indexing: we investigate properties deriving from finite isometric embeddings of a metric space into Euclidean spaces, and we show how these proprieties can be used to perform space pruning, exact search and dimensionality reduction. **Chapter 6** summarizes our achievements and suggests some future research directions.

CHAPTER 2

Background

Contributions presented in this thesis are all related to Content-Based Image Retrieval and Similarity Search. This chapter reviews key concepts as well as many works related to these research fields. It is organized as follows. Content-Based Image Retrieval is introduced in Section 2.1. Then Section 2.2 focuses on image representations for content-based search. Section 2.3 gives a brief overview of recent deep learning approaches and then focuses on Convolutional Neural Networks. Section 2.4 moves to similarity search with a particular focus on metric search and indexing. Finally, Section 2.5 describes datasets used in this thesis.

In Table 2.1 we outline where the various topics serve as a background to the rest of this thesis.[1]

2.1 Content-Based Image Retrieval

Content-Based Image Retrieval (CBIR) embraces any technology that allows systems to organize archives containing digital pictures so that they can be searched and retrieved by using their visual content [84]. So, "content-based" means that the search is performed without using text, tags or other metadata associated with the images but rather analysing the images themselves. This implies that any image can be used as a query (*query-by-example*) and that the retrieval system is expected to returns "similar" database images. This search paradigm lends itself to explore collections of untagged or poorly annotated images, which are nowadays rapidly increasing due to widespread access to digital cameras and the Web.

The content-based search is based on extracting pieces of information from the images, which we call *features*, and on organizing and comparing images on the basis of these features. The request is that images with similar contents are described by features that are mathematically similar. However, teaching systems to automatically extract meaningful information form what is just a grid of coloured pixel is a challenging task, principally due to the *semantic* [236] and *intention* [272] *gaps*. The semantic gap refers to the discrepancy between the limited descriptive power of image features and the richness of high-level semantic concepts that a user can naturally identify in an image. The intention gap is even more subtle since it originates from the polysemy of images and the difficulty that a user suffers to precisely express his/her search intent by a query at hand. Users who submit an image such as the one in

[1] We hope this helps the reader to skip sections that would not extend his/her knowledge or that are not related to a chapter of interest.

Table 2.1: *Structure of this chapter, indicating where the various topics serves as a foundation for subsequent chapters.*

Sections	Topics and Reference Notes
2.1 Content-Based Image Retrieval 2.1.1 Retrieval Performance Measures	Introduction to CBIR and to some popular performance measures, including those used in the experiments of **Chapters 3** and **4**.
2.2 Image representations 2.2.1 Local Features 2.2.1.1 SIFT 2.2.1.2 Binary Local Features 2.2.2 Aggregations of Local Features 2.2.2.1 Bag-of-Words 2.2.2.2 VLAD 2.2.2.3 Fisher Vector	Brief overview of techniques for representing image visual content. Some local features (SIFT, ORB, LATCH, A-KAZE) and aggregations of local features (BoW, VLAD, FV) are presented. SIFTs are used in experiments of **Sections 3.1** and **4.1**. The ORB, LATCH, and A-KAZE binary features are used in **Section 3.2**. BoW, VLAD and FV are used in **Chapter 3**. Moreover, VLAD is used in **Section 4.1** and FV serves as a basis for **Appendix A**.
2.3 Deep Learning 2.3.1 Deep Neural Networks and Deep Features 2.3.2 Convolutional Neural Networks	Introduction to deep learning and recently proposed deep features. CNN are then described. CNN features are used in **Chapter 3** and in **Section 4.2**.
2.4 Similarity Search 2.4.1 Metric Space 2.4.1.1 Distance Measures 2.4.2 Intrinsic Dimensionality 2.4.4 Similarity Queries 2.4.4.1 Range Query 2.4.4.2 Nearest Neighbour Query 2.4.3 Efficiency Measures for Exact Search	Introduction to the similarity search problem and metric search. Basic notions of metric space and normed metric space are provided, as well as popular metric functions (including Euclidean, Hamming, Cosine, Jensen-Shannon and Triangular distances), and similarity queries (range and nearest neighbour search). Euclidean, Hamming, Cosine distances and the similarity queries are used practically in the **whole thesis**. Jensen-Shannon and Triangular distances are used in **Chapter 5**.
2.4.5 Space Partitioning 2.4.5.1 Ball Partitioning 2.4.5.2 Excluded-Middle partitioning 2.4.5.3 Generalized Hyperplane Partitioning 2.4.5.4 Voronoi-like partitioning 2.4.6 Pruning Strategies 2.4.6.1 Object-Pivot Distance Constraint 2.4.6.2 Range-Pivot Distance Constraint 2.4.6.3 Pivot-Pivot Distance Constraint 2.4.6.4 Double-Pivot Distance Constraint	Overview of space partitioning techniques that are typically uses by metric indexes (including those later described in **Section 2.4.7**). State-of-the-art pruning strategies are then reported, which are fundamental for excluding regions of the search space in order to reduce the cost of a query evaluation. The Hyperbolic Exclusion (corollary of the double-pivot distance contains) is of particular interest for **Chapter 5** where we define a similar exclusion condition.
2.4.7 Metric Access Methods for Exact Search 2.4.7.1 Vantage Point Tree 2.4.7.2 Generalized Hyperplane Tree 2.4.7.3 Bisector Tree and Monotone Bisector Tree 2.4.7.4 Spatial Approximation Trees 2.4.7.5 AESA and LAESA	Description of several metric access methods that are used in the experiments of **Chapter 5**.
2.4.8 Metric Space Transformations 2.4.8.1 Finite Isometric Embeddings	Definitions of principal metric space transformations, which are at the core of many approximate search methods, such as permutation-based indexing introduced in **Section 2.4.9.2**. Finite isometric embeddings are used in **Chapter 5**.
2.4.9 Permutation-Based Approximate Similarity Search 2.4.9.1 Performance Measures for Approximate Search 2.4.9.2 Permutation-Based Indexing 2.4.9.3 Surrogate Text Representation	Principles of approximate similarity search and permutation-based indexing. Permutation-based approaches are used in **Chapter 4**. The Surrogate Text Representation is used in **Section 4.1**.
2.5 Datasets 2.5.1 Datasets used for Retrieval and Recognition Tasks 2.5.2 Datasets Used for Training 2.5.3 Datasets Used for Similarity Search 2.5.4 Other Datasets	Descriptions of the datasets used in **Chapters 3, 4, and 5.**

Figure 2.1: *Image polysemy and intention gap.*

Figure 2.1 could have a dozen of different intentions: "find some images of the same scene", "find other pictures of a baby lying in the grass", "find images of the same cathedral", "find images with a child wearing a hat", etc.

Decades of research on relevance feedback, image annotation and search, as well as recent deep learning approaches have helped to mitigate these issues [84, 118, 122, 167, 170, 278]. Meanwhile, techniques of CBIR has been extensively used in many web search engines, where images can be searched by using their visual content [2, 5, 14], and on smartphones apps, where information can be obtained by pointing the smartphone camera toward a monument, a painting, a logo [1, 4, 13].

In facts, a specialization of the basic CBIR technology include the techniques of instance retrieval and object recognition [253], where visual content of images is analysed so that objects contained in digital pictures are recognized, and/or images containing specific objects are retrieved.

CBIR involves not only the extraction of visual information from images and their comparison, but other important aspects like user intention analysis, query formation, database indexing, results scoring, search reranking, retrieval browsing. [84] and [277] are excellent surveys on this topic. In this thesis, we mainly focus on image representations for the instance retrieval task and approaches for indexing and searching the extracted image features.

2.1.1 Retrieval Performance Measures

When evaluating a retrieval system we are interested both in the quality of the results and in the computational performance of the system. The ideal case is having high-quality results, with very fast response and low memory occupation. In practice, the applicative scenario determines the preferred trade-off between effectiveness and efficiency. For example, forensic image retrieval requires high accuracy of the results. Applications running on mobile devices, instead, need to provide fast response and run on systems with limited computation resources; in the latter case, high efficiency is of primary interest, sometimes at the expense of some reduction in the quality of the results.

The measure used to evaluate the efficiency of an algorithm depends on the purpose of the algorithm itself. For general image retrieval systems, usually, it is the time span between receiving the user's query and showing the retrieved results to the user. For feature extraction algorithms, most important factors are the time needed to extract the features and the memory overhead. For an index algorithm, the focus may be on time, the number of needed "operations" (distance computations, disk accesses, etc.), or the amount of memory required to store the index.

As concerning the effectiveness, in this thesis, we focus on evaluating the quality of a list of search results, ordered by their similarity to the query. We follow the standard approach of using a ground-

truth (typically, manually-generated), which containing some representative queries and the lists of the corresponding relevant results. In this context, the two most basic and popular measures are the *precision* and the *recall*. The *precision* is the fraction of retrieved objects that are relevant to the query, so it gives a measure of how exact an algorithm is in returning relevant objects. The *recall* (or *sensitivity*) is the fraction of relevant objects that are successfully retrieved, so it gives a measure of how complete an algorithm is in returning relevant objects.
Formally,

$$precision = \frac{|retrieved \cap relevant|}{|retrieved|} \tag{2.1}$$

and

$$recall = \frac{|retrieved \cap relevant|}{|relevant|} \tag{2.2}$$

where $|\cdot|$ denotes the size of a set. These notions are often expressed also using the following contingency table

	Relevant	**Non-relevant**
Retrieved	True Positive (TP)	False Positive (FP)
Not Retrieved	False Negative (FN)	True Negative (TN)

as *precision*= $\frac{TP}{TP+FP}$ and *recall*= $\frac{TP}{TP+FN}$. Note that precision and recall are set-based measures, since they do not take into account the rank positions of the relevant objects in the result set. Another common measure to evaluate unranked retrieval result-set is the harmonic mean of retrieval and recall, referred to as *F-score*. Other alternative measures are the *accuracy*= $\frac{TP+TN}{TP+TN+FP+FN}$, the *specificity*= $\frac{TN}{TN+FP}$, and the *false negative rate*= $\frac{FN}{FN+TP}$; see also [117, pp.141]. However, it is worth noting that in the context of a retrieval problem the data is often extremely skewed: a high percentage of the data is non-relevant for a given query. Thus, a system returning only non-relevant objects can appear to perform well according to measures such as accuracy or false positive rate. The advantage of the precision and recall measures, instead, is that they focus the evaluation on the return of true positive (relevant) objects.

In the ranked retrieval context, the evaluation should take into account also the rank position of each returned relevant object. If averaging the precision value obtained every time a relevant image is encountered we obtain the so-called *average precision*. It equals the area under the precision-recall curve and so it takes into account both precision and recall. By computing the average precision for multiple queries and averaging all these values we obtain the *mean Average Precision (mAP)*. The mAP provides an overall measure of effectiveness and so it is commonly used for assessing image retrieval systems. We use the mAP quality measure in the experiments of Chapters 3 and 4. In Section 3.1, we evaluate the effectiveness of several state-of-the-art image representations for the recognition task. In that case, we assess the quality of the results also by means of the probability p of finding an image of the same query object within the first r results. It is defined as

$$p(r) = P(R \leq r), \tag{2.3}$$

where R is the random variable denoting the position of the first relevant image in the ranked result list of a query. For $r = 1$, p equals the accuracy of a classifier that recognizes the query object as the most similar that have been found. For $r > 1$, we estimate the probability $p(r)$ as

$$\frac{1}{N}\sum_{i=1}^{N} [\![r_{q_i} \leq r]\!], \tag{2.4}$$

where N is total the number of tested queries, r_{q_i} is the position of the first relevant image when querying q_i, and $[\![\cdot]\!]$ represents the Iverson bracket, which equals one if the argument is true, and zero otherwise.

It is worth noting that, even if not used in this thesis, there are many other measures for evaluating ranked retrieval results, such as *Position Error* and *Cumulative Gain* (see for example [176, 271]).

Figure 2.2: *Images compared by matching their local features and searching for a geometric transformation that can associate the regions of both images. In this example, ORB [221] features and the RANSAC [102] fitting model were used. Note that no features are localized in flat regions (like the sky) of the images because features need to be distinctive. In the right-hand figure, the green lines are two of the four edges of a bounding box (estimated via RANSAC) delimiting the region of the figure that matches the left-hand figure. The other two edges of the bounding box fall outside the image.*

2.2 Image representations

At the core of any CBIR system there is the problem of representing images in some kind of feature space. This means that each image is represented by a one or more features that capture some image characteristics. Image features are typically represented by numerical descriptors that lie in a space where a (di)similarity measure is provided to assess the (di)similarity between a pair of images on the basis of their descriptors.

The image features are expected to be descriptive and discriminative. The general request is that they are invariant, or at least robust, to various image transformations such as translation, rotation, scale changes, illumination changes, etc. Since the computation of a good image representation is crucial to the image retrieval problem, as well as to many computer vision applications, the research for effective image features has been object of much interest from the research community. In the early years of CBIR, *global features* were used to describe the image visual content by means of colour [153, 175], texture [89, 202] and shape [175, 181], to name but a few. The main advantage of global features is that they usually are fast to be extracted and compared; however, they have low discriminative power and are considered too rigid to represent images since they often fail to identify salient visual characteristics.

The intuition that each local region of an image brings a different amount of information had led to the ascendancy of *local features* (Section 2.2.1), like SIFT [173] and SURF [47]. Each local feature provides a description of a local region of an image that is a patch of pixel surrounding an *interest point*. The interest points are typically selected in regions with high-intensity variation (such as corners and blobs) so that they can be repeatability identified under different views of the same image/object. In most of the cases, each image is represented by hundreds or thousands of local features. In order to decide that two images match, since they contain the same or similar objects, local descriptors in the two images need to be pairwise compared to identify matching patterns. Then candidate matches can be validated with a geometric consistency check, *e.g.* by using RANdom SAmple Consensus (RANSAC) [102] that is a de-facto fitting model in computer vision (Figure 2.2).

The image comparison relying on local features suffers of low efficiency. In the worst case, given an image as a query and a database of images, we have to compare each local feature of the query against all the local features of any dataset image. Even if efficient data structures such as kd-tree [103] are used to search matching features, the image comparison based on local descriptors does not scale on large collections. To overcome this issue, researchers have investigated the use of statistical summaries of all the local features extracted from an image in order to encode them into a compact descriptor. *Aggregation techniques* (Section 2.2.2), such as Bag-of-Words [233], VLAD [143] and Fisher Vector [206], revealed to be particularly suitable for image retrieval and search task. In fact, on one hand, they have proved to be effective image representations that still preserve the discriminative power of the local descriptors

[144, 208, 209]; on the other hand, they lead to reduce the cost of image search since each image is represented by a single descriptor rather than thousands. The use of both aggregation approaches and index structures for approximate similarity search (see Section 2.4) guarantees scalability on very large datasets.

In the last few years, image features computed using deep learning (Section 2.3) have emerged as effective image representations, clouding descriptors built upon local features and other hand-crafted descriptors. In fact, the so-called *deep features* achieved the state-of-the-art results in several vision tasks, including image retrieval and object recognition.

2.2.1 Local Features

A local feature is the characterization of a local image pattern that differs from its immediate neighbourhood [250]. The image properties used to identify and describe those local patterns are various and depend on the used approach. The idea at the core of quite all the techniques is to localize some interest points, also called *keypoints*, and then describe a neighbourhood of each of them. The first phase is referred to as *feature detection* and aims to automatically find distinctive points in the images; the latter is the *feature description* that for each keypoint provides a numerical representation of some visual characteristics of a patch surrounding that point.

As highlighted by Tuytelaars et al. [250], ideal local features should satisfy the following properties:

- *repeatability*: invariance and robustness to major image transformations,
- *distinctiveness*: high intensity variation in the patterns surrounding the feature,
- *locality*: description of local patch,
- *quantity*: sufficiently large number of features per image,
- *accuracy*: feature should be accurately identified in the image location, scale and shape,
- *efficiency*: fast feature extraction.

Most of the feature detectors identifies the keypoints at *corners* (*e.g.* Harris [123], SUSAN [237] and FAST [220]), *blobs* (*e.g.* Hessian [48], Difference-of-Gaussian [173], SURF [47], KAZE [16]) or *regions* (*e.g.* MSER [177]) of the image. Usually, not only the location of the keypoint is stored, but also the scale and the orientation at which the feature is detected. Then, for each keypoint, one or more numerical descriptors are computed by using histogram of oriented gradients (SIFT [173], SURF [47]), gradient location and orientation (GLOH [185]), or pixel intensity comparisons (BRIEF [61], ORB [221]), to name but a few.

More details on local features used in this thesis are given below.

2.2.1.1 SIFT

Nowadays, the *Scale-Invariant Feature Transform* (SIFT) [173] is the most cited and used local feature thanks to its distinctiveness for recognition tasks and robustness to several image transformations. It uses a scale-space representation created by a Gaussian pyramid algorithm. Candidate keypoints are localized both in space (2D position in the image) and scale (a level on the pyramid) by searching for scale-space extrema in the Difference-of-Gaussian function convolved with the image. Then keypoints that have low contrast (point sensitive to noise), or that are poorly localized along an edge, are discarded. In order to achieve rotational invariance, each stable keypoint is associated with one or more orientations, estimated on the basis of local image gradients direction. The traditional choice is taking the highest peaks in a histogram of 36 bins that cover the 360-degree range of orientations. Finally, for each keypoint (now including a point, a scale, and an orientation) a descriptor is assigned after transforming the image according to the detected scale and orientation. The feature descriptor is computed as the normalized histogram of orientations of local image gradients (measured in a region around the selected point). Original SIFT uses 4x4 arrays of histograms with 8 orientation bin in each. The final feature vector is thus of 4x4x8=128 dimensions.

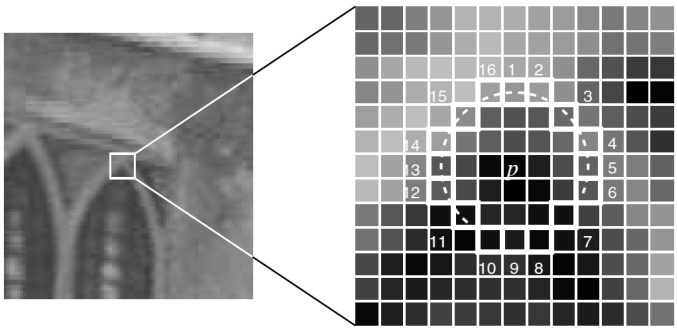


Figure 2.3: *Visualization of the FAST keypoint detector reported from [220].*

The overall SIFT extraction is costly due to the local image gradient computations. In [47] integral images were used to speed up the computation and the *Speeded Up Robust Features* (SURF) was proposed as an efficient approximation of the SIFT.

2.2.1.2 Binary Local Features

Binary local features [15, 61, 165, 221] were recently introduced to fulfil the need of compact local descriptors that are fast to be extracted and compared. Binary descriptors are, as the name suggests, binary strings with low memory footprint (usually 128, 256 or 512 bits). They are compared using the Hamming Distance [121] (see Section 2.4.1) which is extremely efficient to be computed using the bitwise XOR operation. However, the fundamental characteristic of these approaches is not just using a "binary representation" (that can be obtained from any image feature by means of hashing or binarization algorithms), but rather efficiently derive the binary descriptors from the image itself. To this end, researchers have used pixel intensity comparisons, which are notably faster than (approximate or exact) local gradients used in most state-of-the-art local features, like SIFT and SURF.

Nearly all the binary descriptors proposed in literature involve the following steps:

1. sample some pixel or patches of pixel in a region around a keypoint;
2. use a mechanism to measures the orientation of the region describing the keypoint and rotate it;
3. select a set of pixel or patch tuples (*e.g.* pairs, triples, etc.), and for each tuple compute a bit value (0-1) as the result of some comparisons between the objects of the tuple.

Therefore, the main difference between the various approaches relies mainly on the choice of the point/-patch sampling pattern, the orientation mechanism, and the pixel/patch comparison rules. For example, the selection of the pixel tuples can be performed either randomly (BRIEF [61]), using a hand-crafted pattern (BRISK [165]) or automatically leaned from training data (ORB [221], FREAK [15], LATCH [166]).

In the following we briefly describe some popular binary features, that we also use in the experiments of Section 3.2.

ORB The *Oriented FAST and Rotated BRIEF* (ORB) [221] descriptor, was built upon the well know FAST [220] keypoint detector and the BRIEF [61] binary descriptor.

The original FAST uses fast pixel comparisons to identifies corner in images: as showed in Figure 2.3, it considers a circle of sixteen pixels around a candidate corner p. Then it classifies the point p as a corner if there exist 12 contiguous points that are brighter than p by more than a threshold. However, analysing only the four pixels at 1, 5, 9 and 13 permit to efficiently exclude many non-corner points so that the full segment test criterion is applied only to the remaining candidates.

ORB adds a rotation component to FAST by using a simple measure of corner orientation based on local first-order moments. Then for each corner (keypoint) it computes a binary descriptor as done in

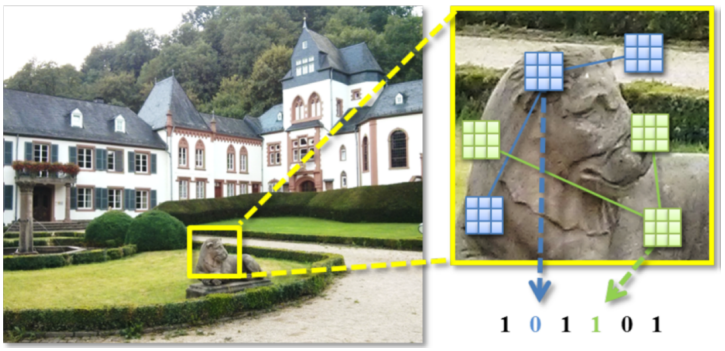


Figure 2.4: *Visualization of the LATCH descriptor reported from [166]. Each bit of the LATCH descriptor is computed by comparing a triplet of pixel patches.*

BRIEF, which is the first, and also the simplest, of the family of binary descriptors. BRIEF is based on intensity comparison of *pairs* of *pixels* randomly selected in a patch centred around a keypoint. So given a set $\{s_1, \ldots, s_D\}$ of pixel pairs, BRIEF computes a binary vector of length D, where the i-th bit is the result of an intensity comparison over the i-th pair. Specifically, given the pixel pair $s_i = (p_{i,1}, p_{i,2})$, if the intensity (smoothed by a Gaussian filter) of the pixel $p_{i,1}$ is greater than that of $p_{i,2}$, then the i-th bit of the binary descriptors is set to 1; otherwise, it is set to 0. ORB, unlike BRIEF, uses an unsupervised learning approach to select the pixel pairs (rather than random selection).

Interestingly, the total ORB extraction process (keypoint and orientation detection + feature description) is an order of magnitude faster than SURF, and two orders faster than SIFT, according to the experimental results reported in [126, 186, 221].

LATCH Comparing single smoothed intensities of pixels, as done in many binary descriptors, may be sensitive to noise or slight image distortion. To overcome this issue Levi et al. [166] proposed to use small *patch* comparisons, rather than pixel comparisons, and *triplets* rather than pairs of sampling points. The resulting image descriptor was named *Learned Arrangements of Three patCHes codes* (LATCH).

In [166], the authors considered windows of 48×48 pixels centred on precomputed keypoints. Then, for each image window, a certain number D of patch triplets are selected (*e.g.* $D = 512$) using a supervised learning approach. The original LATCH uses patches of 7x7 pixels whose similarity is measured by the sum-of-squared-differences (SSD). Each triplet is composed by one "anchor" patch (a) and two "companion" patches (c_1, c_2). If for the i-th triplet $SSD(a, c_1) < SSD(a, c_2)$ then the i-th bit of the LATCH descriptor is assigned to 1. If not, it is assigned to 0. Figure 2.4 exemplified the computation of LATCH descriptor using triplets.

A-KAZE In [17], Alcantarilla et al. proposed a fast feature detection and description approach, called *Accelerated-KAZE* (A-KAZE). It can be considered a SIFT-inspired method since it uses a multiscale approach. It use a *Fast Explicit Diffusion* method [116] to build a non-linear scale space. Then, keypoints are detected by searching maxima in scale and spatial location. The descriptors are then built using information carried out from the non-linear scale space and using binary test between the average of areas around keypoints instead of single pixels. Interestingly, while the Gaussian blurring used to build the scale space in SIFT does not preserve the object boundaries, the non-linear diffusion filtering technique used in A-KAZE preserves edges [213].

2.2.2 Aggregations of Local Features

In this section we describe some leading approaches to encode a set of local descriptors into a fixed-length representation. By far, the most popular aggregation method has been the Bag-of-Words (BoW) [233] (Section 2.2.2.1). It uses a finite *visual vocabulary* to quantize the local descriptors extracted from images and then represents each image as a histogram of occurrences of visual words. The BoW representation is very sparse and can be efficiently index by using inverted files.

In the last decade, other effective encoding schemes, namely the Vector of Locally Aggregation Descriptors (VLAD) [143] (Section 2.2.2.2) and the Fisher Vectors (FVs) [206] (Section 2.2.2.3), have been widely used for both image classification and large-scale image search. The VLAD method, similarly to BoW, quantizes the local descriptors of an image by using a visual vocabulary. Differently from BoW, VLAD encodes the accumulated difference between the visual words and the associated descriptors, rather than just the number of descriptors assigned to each visual word. So, it exploits more aspects of the distribution of the local descriptors of an image.

The FV uses the Fisher Kernel framework [137] to transform an incoming set of descriptors into a fixed-size vector representation. The basic idea is to characterize how the sample of descriptors deviates from an average distribution that might be understood as a "probabilistic visual vocabulary". As highlighted in [144], the FV can be viewed as a probabilistic version of the VLAD. Moreover, compared to BoW, which takes into account just 0-order statistics (occurrences of visual words), the FV offers a better representation by encoding higher order statistics (first and optionally second order) related to the distribution of the descriptors. Both FV and VLAD uses smaller visual vocabulary than BoW to achieve a given performance so they result in a more efficient representation. However, the final VLAD and FV descriptors are not sparse, inverted file scheme is not suitable index images based on these encodings. Typically, techniques of dimensionality reduction, such as the Principal Component Analysis (PCA)[2] [50, 148], compression with product quantization [115, 142] and binary codes [208] are used to efficiently store these features. Moreover, multi-dimensional index or indexing methods for similarity searching must be used to scale up to large datasets.

2.2.2.1 Bag-of-Words

The *Bag of (Visual) Words (BoW)* was initially proposed in [233] for matching objects throughout a video database. Thereafter, it has been widely used for classification and CBIR tasks [81, 141, 211].

BoW uses a "visual vocabulary" to group together the local descriptors of an image and represent each image as a set (bag) of visual words. The visual vocabulary is learned by clustering a large set of local descriptors extracted from training images. The most common choice is using the k-means [171] or the hierarchical k-means [195]. The cluster centres (*centroids*) are regarded as the "visual words" of the vocabulary. Then, each local feature of an image is associated to its closest centroid and the image is represented by a histogram of occurrences of visual words (Figure 2.5).

This approach was inspired by the BoW model used in natural language processing and information retrieval [224], thus many text indexing techniques, such as inverted files [263], have been applied for image search. In such cases the retrieval phase is performed by using the visual words in place of text words and considering a query image as disjunctive term-query. However, as highlighted in [273], an image query contains much more terms than a text query, e.g. 1500 visual terms rather than 3 text terms. Therefore, thousands of posting lists should be accessed in the visual case.

From the very beginning [233] some words reduction techniques were used (e.g. removing 10% of the more frequent words) and images have been ranked using the cosine similarity measure (Section 2.4.1) in conjunction with the standard *term frequency-inverse document frequency (tf-idf)* [224] weighting. In order to improve the efficiency of BoW, several approaches for the reduction of visual words were investigated [25, 242]. Moreover, search results obtained using BoW were also improved by applying re-ranking approaches [68, 139, 211, 246] and exploiting additional geometrical information [205, 211, 244, 275]. In fact, the BoW encoding can additionally store the location of the local features associated to each visual word in order to fast marching the local features of two images. In this case, any two local features assigned to the same visual word are considered to match. Geometry consistency check like RANSAC [102] can be further applied.

The quantization process used in the baseline BoW introduces a loss of information about the original descriptors. For example, corresponding descriptors in two images may be assigned to different visual words. To overcome the quantization loss, more accurate representation of the original descriptors

[2] The PCA is the most popular of the techniques for unsupervised dimensionality reduction. The idea is to find a linear transformation of n-dimensional to k-dimensional vectors ($k \leq n$) that best preserves the *variance* of the input data. Specifically, PCA projects the data along the direction of its first k principal components, which are the eigenvectors of the covariance matrix of the (centred) input data.

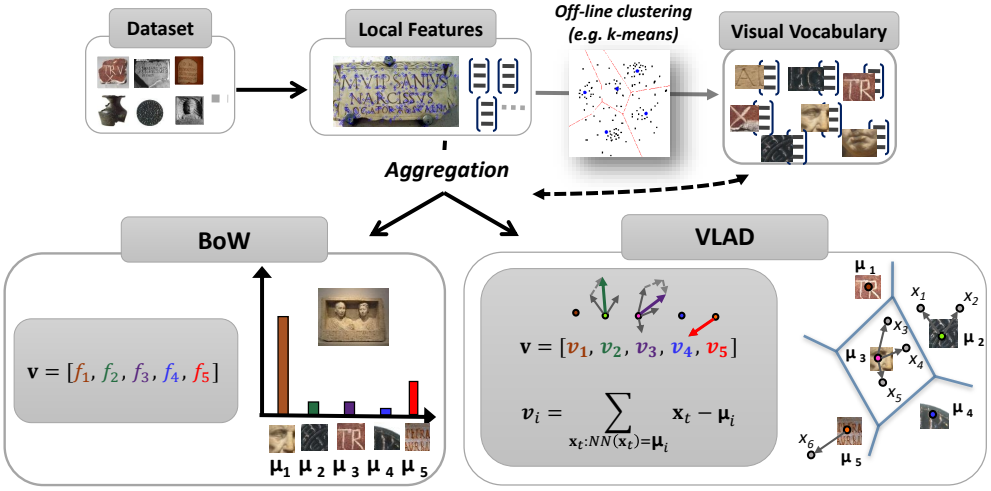


Figure 2.5: *Simplified illustration of BoW and VLAD encodings. Given a visual vocabulary and the local features extracted from an image, the BoW encodes the number of descriptors assigned to each visual word while the VLAD encodes the accumulated differences between each visual word ($\boldsymbol{\mu}_i$) and the associated descriptors ($\mathbf{x}_i$)*

and alternative encoding techniques have been used, such as Hamming Embedding [139, 141], soft-assignment [212, 255, 256], multiple assignment [141, 145], locality-constrained linear coding [260], sparse coding [55, 265] and the use of spatial pyramids [158].

2.2.2.2 VLAD

The *Vector of Locally Aggregation Descriptors (VLAD)* is an encoding scheme introduced in [143]. It adopts the k-means to build a visual vocabulary $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$, also called "codebook". Similarly to BoW, each local feature ($\mathbf{x}_t$) of an image is associated its closest visual word ($NN(\mathbf{x}_t)$) in the codebook. Then for each visual word, VLAD accumulates the *residual vectors*, each defined as the differences $\mathbf{x}_t - \boldsymbol{\mu}_i$ between the centroid $\boldsymbol{\mu}_i$ and the local feature $\mathbf{x}_t$ assigned to it (Figure 2.5). Formally, the accumulated residual vector for the centroid $\boldsymbol{\mu}_i$ is

$$\mathbf{v}_i = \sum_{\mathbf{x}_t : NN(\mathbf{x}_t) = \boldsymbol{\mu}_i} \mathbf{x}_t - \boldsymbol{\mu}_i. \tag{2.5}$$

Finally, the vectors $\mathbf{v}_i$ are concatenated into a single descriptor $\mathbf{v} = [\mathbf{v}_1, \cdots, \mathbf{v}_K]$ referred to as VLAD.

Throughout the thesis, we refer to the accumulated sub-vectors $\mathbf{v}_i$ simply as "residual sub-vectors". All the residual sub-vectors have the same size D which is equal to the size of the used local features. Thus the dimensionality of the whole vector V is fixed and equals DK. Relatively small number of centroids (*e.g.* $K = 64 - 256$) is used.

A power-law normalization ($\boldsymbol{v} \rightarrow |\boldsymbol{v}|^{\beta}\text{sign}(\boldsymbol{v})$) and a ℓ_2-normalization ($\boldsymbol{v} \rightarrow \boldsymbol{v}/\|\boldsymbol{v}\|_2$) are usually applied.[3] After this two VLADs can be effectively compared using the Euclidean distance or, equivalently, the inner product [41, 144].

Since VLAD descriptors have high dimensionality, PCA can been used to obtain a more compact representation [143]. In literature, several enhancements to the basic VLAD were proposed [41, 67, 87, 275].

[3] A common choice for the exponent β is 0.5.

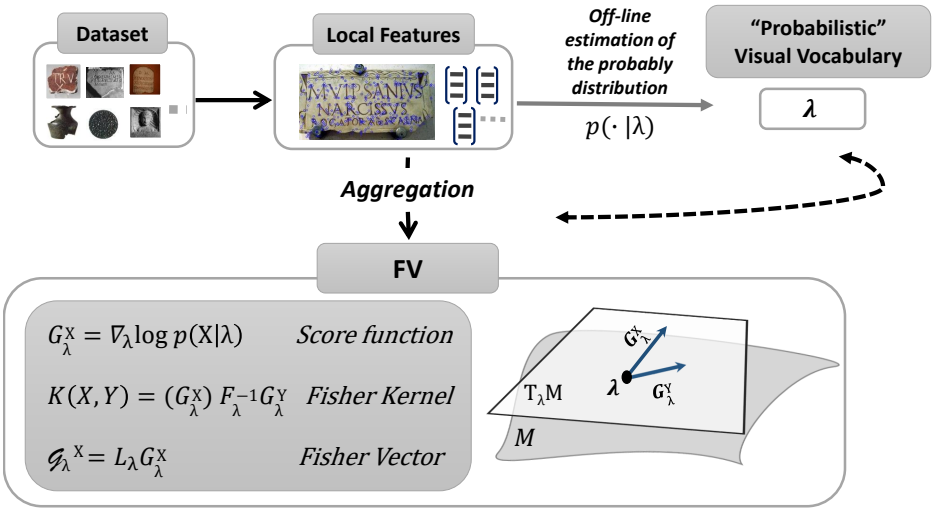


Figure 2.6: *Simplified illustration of the FV encoding. The family of distribution $M = \{p(\cdot|\lambda)\}_\lambda$ parametrized by a vector λ can be regarded as a Riemannian manifold with a local metric given by the FIM (F_λ) [227]. Given a set X of local descriptors of an image we can compute the score function G_λ^X which lives in the tangent space $T_\lambda M$ and give us the direction in which parameter λ should be modified to best fit X. The distance between two score functions is given by the metric induced by F_λ that can be expressed in term of Euclidean distance between the corresponding FVs.*

2.2.2.3 Fisher Vector

The Fisher Kernel is a powerful framework introduced in [137] for classifying DNA splice site sequences and to detect homologies between protein sequences. In [206], the Fisher Kernel was adopted in the context of image classification as an efficient tool to encode image local descriptors into a fixed-size vector representation.

The main idea of this method is to derive a kernel function to measure the similarity between two sets of data, such as the sets of local descriptors extracted from two images. Specifically, the similarity of two sample sets X and Y is measured by analysing the difference between the statistical properties of X and Y, rather than comparing directly X and Y. To this scope a probability distribution $p(\cdot|\lambda)$ with some parameters $\lambda \in \mathbb{R}^m$ is first estimated on a large training set and is used as "average distribution" over the space of all the possible data observations. Then each sample $X = \{x_1, \ldots, x_T\}$ of data observations is represented by a vector, named *Fisher Vector*, that indicates the direction in which the parameter λ of the probability distribution $p(\cdot|\lambda)$ should be modified to best fit the data in X. In this way, two samples are considered similar if the directions given by their respective Fisher Vectors are similar.

Specifically, as proposed in [137, 206], the similarity between two sample sets X and Y is measured using the Fisher Kernel, defined as

$$\mathcal{K}(X,Y) = (G_\lambda^X)^\top F_\lambda^{-1} G_\lambda^Y, \tag{2.6}$$

where F_λ is the Fisher Information Matrix (FIM) and $G_\lambda^X = \nabla_\lambda \log p(X|\lambda)$ is the *score function*.

The computation of the Fisher Kernel is costly due to the multiplication by the inverse of the Fisher Information Matrix (FIM). However, by using the Cholesky decomposition $F_\lambda^{-1} = L_\lambda^\top L_\lambda$, it is possible to re-written the Fisher Kernel as an Euclidean dot-product, *i.e.*

$$\mathcal{K}(X,Y) = (\mathcal{G}_\lambda^X)^\top \mathcal{G}_\lambda^Y, \tag{2.7}$$

where

$$\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X \tag{2.8}$$

is the FV of X (Figure 2.6). Note that the dimensionality of the FV depends only on the dimensionality m of the parameter λ.

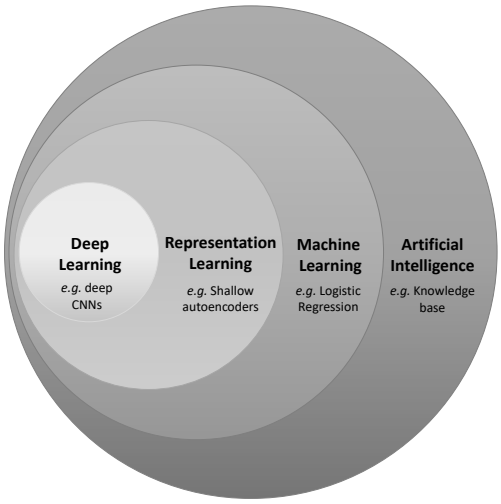


Figure 2.7: *A Venn diagram showing the relationship between Deep Learning, Representation Learning, Machine Learning and Artificial Intelligence. Figure adapted from [112].*

The FV is usually divided by $T = |X|$ in order to avoid the dependence on the sample size [226] and ℓ_2-normalized because, as proved in [209, 226], this is a way to cancel out the fact that different images contain different amounts of image-specific information (*e.g.* the same object at different scales).

In the original definition of the FV [206, 208] the distribution $p(\cdot|\lambda)$ is chosen to be a Gaussian Mixture Model (GMM) of parameter

$$\lambda = \{w_k, \boldsymbol{\mu}_k = [\mu_{k1}, \ldots, \mu_{kD}], \boldsymbol{\Sigma}_k = \text{diag}(\sigma_{k1}, \ldots, \sigma_{kD})\}_{k=1,\ldots,K}$$

where K is the number of Gaussian, D is the dimension of each local descriptor, and w_k, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ are the mixture weight, the mean vector and the covariance matrix of the k-th Gaussian, respectively. By using the GMM model, the FV of a set of D-dimensional local descriptors $X = \{x_1, \ldots, x_T\}$ is obtained as the concatenation of the vector $\mathcal{G}^X_\alpha \in \mathbb{R}^K$, $\mathcal{G}^X_{\boldsymbol{\mu}} \in \mathbb{R}^{KD}$, $\mathcal{G}^X_\sigma \in \mathbb{R}^{KD}$, computed as

$$\mathcal{G}^X_{\alpha_k} = \frac{1}{T\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k) \qquad k = 1, \ldots, K \tag{2.9}$$

$$\mathcal{G}^X_{\mu_{kd}} = \frac{1}{T\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \frac{x_{td} - \mu_{kd}}{\sigma_{kd}} \qquad k = 1, \ldots, K,\ d = 1, \ldots, D \tag{2.10}$$

$$\mathcal{G}^X_{\sigma_{kd}} = \frac{1}{T\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \frac{1}{\sqrt{2}} \left[\frac{(x_{td} - \mu_{kd})^2}{(\sigma_{kd})^2} - 1 \right] \qquad k = 1, \ldots, K,\ d = 1, \ldots, D \tag{2.11}$$

where $\gamma_t(k) = p(k|x_t, \lambda)$ is the probability for the observation x_t to be generated by the k-th Gaussian. The whole FV is of dimension $(2D + 1)K$. However, the FV is often used considering only the sub-vector associated with the mean parameters ($\mathcal{G}^X_{\boldsymbol{\mu}}$) whose dimensionality is KD [143, 144, 208]. PCA can also be applied to reduce the dimensionality.

2.3 Deep Learning

Recent years have witnessed an explosion of interest in a branch of machine learning called *deep learning* (Figure 2.7). Deep learning is a class of "*representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract*

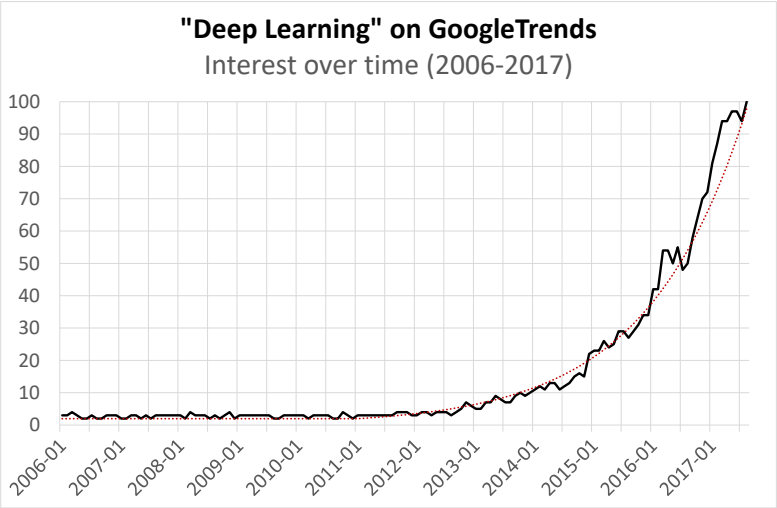


Figure 2.8: *Interest over the time of the term "Deep Learning" estimated using Google Trends (`https://trends.google.com`). Numbers represent search interest relative to the highest point on the chart for the considered time interval. A value of 100 is the peak popularity for the term, a value of 50 means that the term is half as popular, and so on.*

level" [161]. Having those multiple levels of representation/abstraction is fundamental since it gives generalization by allowing systems to automatically learn a hierarchy of features and deal with new domains without having previous knowledge of them. Each level of a deep models depend on a certain number of parameters that are learned from data by using general-purpose learning procedure, which can be supervised, semi-supervised or unsupervised. Typically, a deep model has hundreds of millions of trainable parameters and so need hundreds of millions of training data [161].

Deep learning origins date back to the 1940s-1960s [125, 179, 219] and the 1980s-1990s [106, 160, 162, 222]. It was relatively unpopular for several years until 2006, when seminal works of research groups led by Geoffrey Hinton (University of Toronto), Yoshua Bengio (University of Montreal) and Yann LeCun (New York University) [49, 130, 216] have started to attract the attention of the research community. However, it is only from 2012 that the popularity of deep learning has started to grow exponentially (Figure 2.8).

What happened in 2012? Deep learning approaches achieved record-breaking results in speech recognition [82, 188] and image classification [155]. To give an idea we report the image case. In September 2012, these was a famous competition (ILSVRC2012) with the goal of estimating the content of images for retrieval and image annotation using a subset of the large hand-labelled ImageNet dataset [88] as training. There were three tasks: classification, classification with localization and fine-grained classification.[4] Leading groups in computer vision participated to this challenge. They manly used hand-engineered features, like Fisher Vector encodings of local features, and most of them managed to get an error between 26% and 27% on the classification task. Also the University of Toronto participated at the competition with a team composed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Differently from all the other teams, they used a deep learning approach. Their deep Convolutional Neural Network (named AlexNet [155]) got an error rate of 16%, which created a huge gap in the competition results. The next year, 24 teams participated to the ILSVRC2013 competition. Of these only three used a non-deep learning approach, which was an early sign of the imminent "Cambrian explosion" [215] of the deep learning technologies. In fact, deep learning have recently changed the research landscape in a broad set of domains, not only image classification [155], but also visual object recognition [217], speech recognition [129], natural language processing/understanding [71], recommendation systems [223], bioinformatics [164].

[4] see `http://www.image-net.org/challenges/LSVRC/2012/` for further details

While the state-of-the-art results obtained using deep learning are the key to its success, three main factors have created a breeding ground for the resurgence of deep learning approaches: i) availability of large datasets to train the models; ii) cheap and fast GPUs (to train and run large models); iii) algorithm improvement with respect to the early works. Moreover a lot of open-source libraries (like Caffe, Theano, Torch, TensorFlow, PyLearn2, MXNet) and cloud services (like Amazon Web Services and Microsoft Azure) have supported many research or industrial projects on deep learning.

The Convolutional Neural Network (CNN) (Section 2.3.2) is just one kind of deep leaning models, which is mainly used to deal with images. Other examples of deep models are Recurrent Neural Networks (RNN) (*e.g.* suitable to learn sequences) and Deep Autoencoders (*e.g.* suitable for data compression), just to name a few.

2.3.1 Deep Neural Networks and Deep Features

A deep neural networks is an artificial neural network composed of *multiple layers*: an input layer, some hidden layers, and an output layers. Each hidden layer computes certain mathematical operations in order to extract some knowledge from its input and generate an output, which is then taken as input by the next layer, and so on. The term "deep" means that we have more than one hidden layer. The hidden layers should include some *non-linear operations*, otherwise we could reduce the network to a "shallow" one. The operations made in each layer are parametrized by some *learnable weights*. The most common case is learning the parameters by using a supervised algorithm that optimize a certain objective function (*e.g.* minimizing a loss function). Once the network is trained it can be used to extract the desired information from a set of previously unseen data. The kind of output produced by the network depends on the task addressed, *e.g.* classification, regression, metric learning, etc. Interestingly, in order to produce the final output the deep neural network extracts a hierarchy of features (*deep features*), which are the output produced at each hidden layer. The key aspect is that these features are extracted using the parameters that the network learned directly from the training data, and so that are not designed by human engineers. In a feed-forward deep neural network, that is a deep neural network where the connections between the units of the layers do not form a cycle, the features extracted in the first layers are less abstract than that extracted in the last (higher) layers, which are also more specific for the task on which the network has been trained. This allows us to use pretrained models to extract features to be then used for tasks different from (but related to) the one used to train the network. For example, the famous AlexNet [155] was trained in an image classification scenario, however the features extracted from its intermediate layers have been proved to be effective descriptors of the image visual contents for image retrieval and recognition tasks [39, 217, 268]. This is just an aspect of the *transfer learning* that is made possible by deep models. Another aspect that goes in this direction is the *fine-tuning* of pretrained model: A model pretrained on a specific task and dataset can be re-trained to deal with different task or data domain by somehow re-use the prelearned parameters of the model. For example, some of the parameters of the model are frozen before the new training phase (where eventually some of the last layers are replaced by new ones), or they are used for initialize the model, instead of randomly generated values, in conjunction with a small learning rate. On of the main advantage of fine-tuning is that it led to "train" (or better "adapt") a deep model even if using a relatively small training dataset.

2.3.2 Convolutional Neural Networks

Let us now introduce the CNNs that are feed-forward neural networks particularly suitable for data that has a grid-like topology such as image data or time-series data. A CNN takes as input a tensor, which is a multidimensional array, and will output a high-dimensional structured object, *e.g.* the probabilities that the input data belongs to a certain class for a classification task, real values for regression task, etc. For example in the case of image classification, the network takes as input a coloured image (that is a three-dimensional tensor, where the last dimension refers to the RGB values) and output a vector of scores (*e.g.* 0.70 for cat, 0.10 for tiger, 0.05 for dog, etc.).

According to the definition given by [112], a deep CNN is a deep neural network that uses a *convolution* in at least one of its layers.[5] In general, the convolution is an operation (indicated with $*$) of two

[5] Thereafter we use the term CNN to indicate *deep* CNN, *i.e.* we always consider the case of multiple hidden layers.

functions $x(t), w(t)$ that produce a third function

$$s(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(t-\tau)w(\tau)d\tau.$$

Actually, most of the neural network libraries and books uses the *cross-correlation*, defined as

$$s(t) = \int_{-\infty}^{\infty} x(t+\tau)w(\tau)d\tau,$$

but call it "convolution"; thereafter we will do the same. In machine learning terminology, the first argument x is the *input* (or *receptive field*), the second argument w is the *kernel* (sometimes referred to as *filter*), and the output s is the so called *feature map* (or *activation map*). Here we consider the case in which both the input and the kernel are finite-dimensional tensors with discrete indices (*e.g.* $x(t)$ is the value of x at the multidimensional index t, where t assumes a finite set of values). This means that in practise the convolution is computed as a finite sum. Foe example, given x of dimension $n_x \times m_x$, and the kernel w of dimension $n_w \times m_w$, then the feature map is a tensor $(n_x - n_w + 1) \times (m_x - m_w + 1)$ whose element at the index (t_1, t_2) is

$$s(t_1, t_2) = \sum_{\tau_1} \left(\sum_{\tau_2} x(t_1 + \tau_1, t_2 + \tau_2) w(\tau_1, \tau_2) \right), \tag{2.12}$$

where the indexes range according to domains of definition of the considered tensors. Figure 2.9 gives an example of 2D convolution. Note that the standard convolution shrinks by one element less than the kernel width; to control the size of the feature maps, the input is typically zero-padded. Moreover, in the context of neural networks, a stride is used to control how the kernel convolves around the input volume (Figure 2.10). We refer the interested reader to [112] for further details.

Typically, in a CNN there are two kinds of hidden layer: the *convolutional* and the *fully-connected layer*. In most of the cases the convolutional layers are composed of several stages, notable including

- *Convolutions:* several convolutions are performed in parallel to produce the feature maps. It is worth noting that the kernel of a convolution acts as feature detector, which explains the terminology of " feature map" used for the output.
- *Non-linear function:* a non-linear activation function is applied to each element of the map. An example is the Rectified Linear Unit transform (ReLU) that replaces all the negative values by 0 (ReLU$(x) = \max(0, x)$).
- *Pooling:* element at a certain location of the feature map are replaced with a summary statistic of the nearby elements. Popular pooling functions are max-pooling [279] (which output the maximum value within a rectangular neighbourhood) and the Euclidean normalization of a rectangular neighbourhood.

As pointed by Le Cun et al. [161] the role of the convolution stages is to *detect local conjunction of features from the previous layer*, while the pooling modules aim to *merge semantically similar features into one*. In general, the convolutional layers leverage on sparse connectivity (achieved using a kernel smaller than the input), parameter sharing (each member of the kernel is used at every position of the input, excluding boundary pixels), equivariance to translation (if the input is translated, the output change in the same way).

The second popular kind of layer is the fully connected one that, as the name suggests, is a layer where each unit receives information form all the units of the previous layer (full connectivity). It is typically composed by an "inner product" stage, where the input is simply multiplied by a weight matrix, followed by some non-linear stages. Given the large numbers of parameters in the fully-connected layers, some regularization techniques are used to prevent overfitting and improve the performance. For example the dropout technique randomly remove some units from the networks along with their incoming and outcoming connections, which prevent complex co-adaptation on the training data [131].

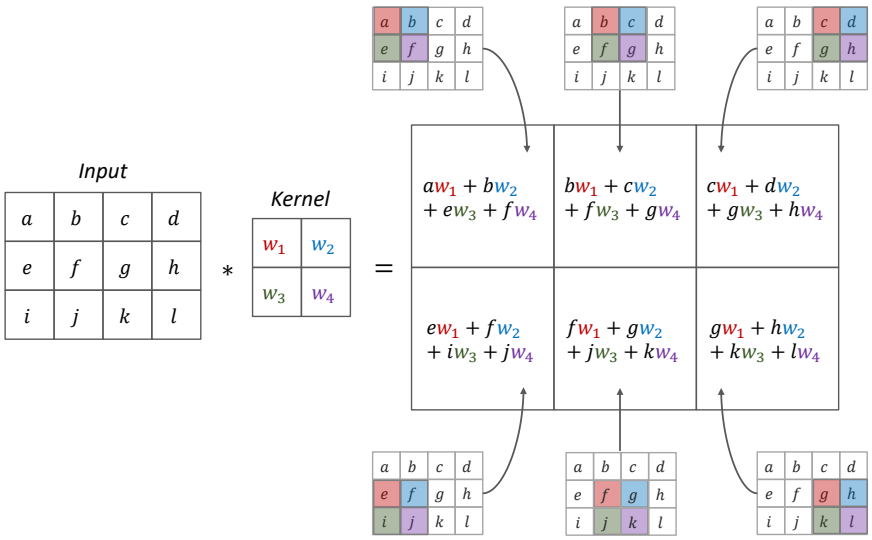


Figure 2.9: *Example of convolution of 2D tensors.*

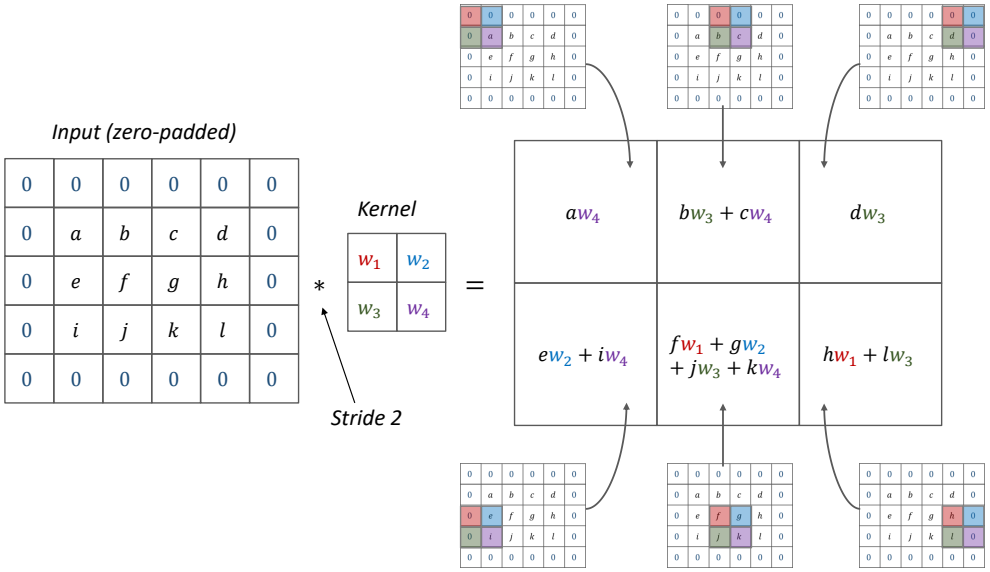


Figure 2.10: *Example of convolution of 2D tensors with a stride of 2 in both the dimensions and zero-padding of the input.*

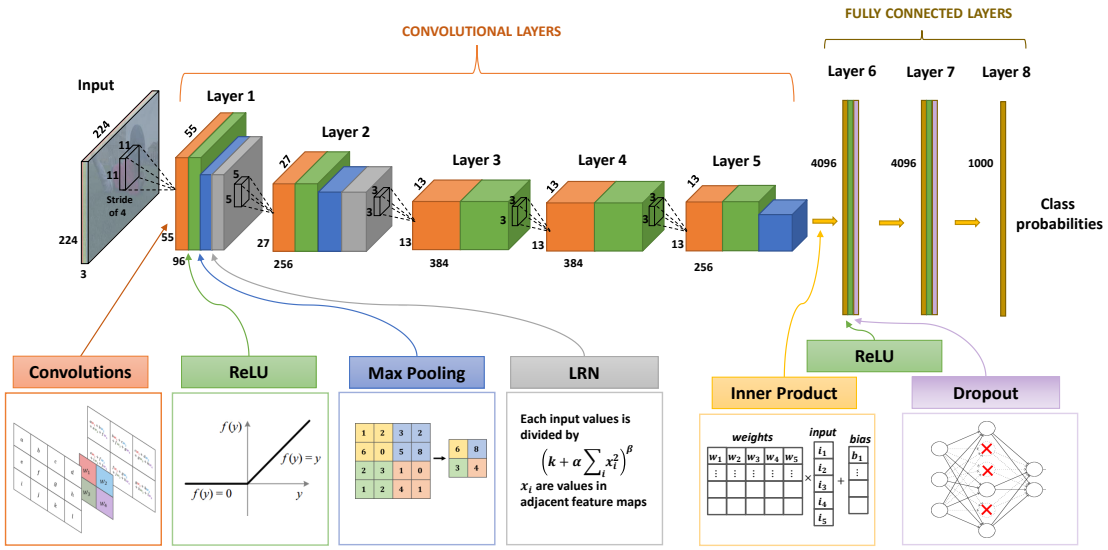


Figure 2.11: *Example of a CNN model: the BVLC Reference CaffeNet. The input is a RGB image of size 224x224 pixels. Orange units corresponds to outputs of convolutions; green units corresponds to outputs of the ReLU; blue units correspond to outputs of max-pooling; grey units correspond to outputs of the Local Response Normalization (LRN); yellows units are the outputs of the fully-connected stages; purple blocks indicates the use of dropout regularizations. The last layer ends with a softmax module producing a distribution over the 1000 training classes.*

Figure 2.11 show a simplified version of the BVLC Reference CaffeNet architecture, which mimics the original AlexNet [155], with minor variations as described in [146].

More advanced and complex networks/modules have been proposed in the last years, notable including Network in Network [169], Inception modules [240], residual networks [124] and several non-linear activation and pooling function variants [187].

CNN Features As we already mentioned, the intermediate outputs (activations) of a feed-forward deep neural networks can be used as data features for generic tasks that may differ from the originally trained task. The first work in this direction using a CNN was [94] where the outputs of the sixth and seventh layers, namely *fc6* and *fc7*, of the pretrained AlexNet, were effectively used for object recognition, scene recognition, domain adaptation, and fine-grained recognition. Babenko et al. [43] used the output of the fifth layer, namely *pool5*, and that of *fc6* and *fc7* for image retrieval. Similarly, [110, 218, 259] demonstrated that the deep features achieves high effectiveness in various vision and retrieval task. Recently, Tolias et al. [247] exploited the activations of the CNN convolutional layers to derive representations for image regions that are then aggregated into a compact vector representation, called Regional Maximum Activation of Convolutions (R-MAC). Moreover, other works started to treat features in a convolutional layer as local features to be then aggregated, using for example VLAD or FV [39, 268].

Finally, it is worth noting that the features extracted from an image by various layers of a CNN capture image characteristics at various level of abstraction. For example, Zeiler and Fergus [270], by visualizing which parts of an input image caused a given activation in the features maps of a model similar to AlexNet, showed that: the output of the first convolutional layer capture the presence or absence of edge at particular orientation and location in the image; the second layer detects corners and other colour/edge conjunction; the third layer mainly capture motifs and textures; the fourth layer is more class-specific and start detecting parts of objects; the last convolutional layer detect entire objects. Features extracted by the fully connected layers, instead, capture more class-related and high-level characteristics.

2.4 Similarity Search

Searching data objects that are close to a given query element under some similarity (or distance) function is a fundamental task in a variety of application domains, including multimedia information retrieval, data mining, pattern recognition and computational biology, to name but a few. This search paradigm, referred to as *similarity search*, overcomes limitations of traditional *exact-match search* that, as pointed in [271, ch.1] *"is neither feasible nor meaningful for data types in the present digital age"* since *"modem digital collections lacks structure and precision"*. In the exact-match scenario, data is searched using a fully comparable key, and the result of a query to a database is the set of objects whose key match the query one. Therefore, exact-match has little meaning when dealing with complex data such as image, audio and video, except for limited applications like searching for exact copies of a query object. The similarity search, instead, is more suitable to search for complex and unstructured data. For example, in the case of CBIR, the search is not computed at the level of the actual images (*e.g.* pixel by pixel comparison) or as exact-match of the image representations, because even a small perturbation of an image may affect the numerical values of its pixel intensities and of the associated image features. The search is rather computed at the level of proximity of the image representations: the images whose representations are closest to the representation of the query are top ranked. There are many other important practical examples captured by the similarity search framework, see, for example, Chávez and Navarro [66] and Zezula et al. [271].

To search a database for the objects similar to a query we can use either a similarity function or a distance function. In the first case, we search for the objects with the greatest similarity to the query. In the latter case, we search for the objects with the lowest distance from the query. A similarity function is said to be equivalent to a distance function if the ranked list of the results to a query is the same.

One main issue related to the similarity search is the *efficiency* since the search algorithms should provide results quickly and with low cost. As we see later the cost highly depends on the function used to compare the objects and how the data is structured and searched. Similarly to [271], we define the distance search problem as follows:

Definition 2.4.1 (Distance Search Problem). Let $\mathcal{D}$ be a domain and $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ a distance measure on $\mathcal{D}$. Given a finite subset $\mathcal{S}$ of $\mathcal{D}$, preprocess or structure the data so that proximity queries are answered efficiently.

The distance function used to compare data objects is chosen on the basis of the application domain and of the nature of the query. Typically a proper distance is considered thanks to the solid mathematical foundations underlying the notion of *metric space* (see Section 2.4.1). In fact, we will see that properties governing the metric space, such as the triangle inequality, can be exploited to index and efficiently search data objects.

In the realm of metric search methods we distinguish between *exact* and *approximate* methods. Given a query, the *exact methods* guarantees to find the true result-set (*e.g.* all the object within a given distance to the query), no matter how much computational resources are required[6]. However, as highlighted by many researchers [64, 190, 261, 271], exact search work well only in space with low *intrinsic dimensionality*[7]. For example, Weber et al. [261] experimentally showed that exact method based on space partitioning degrades to sequential scan when dimensionality exceeds ten. This phenomenon, well know as "*curse of dimensionality*" [210], means that the search scale poorly with data dimensionality and thus a large fraction of the data need to be inspected to answer a query. *Approximate search* methods are less (but still) affected by this phenomena and so they are used to face the search on a large dataset or on space with high intrinsic dimensionality. One common approach is to lose some information and map the data object into a more tractable space in order to perform the search in that space. Of course, the more

[6] Note that an *exact* method is not an *exact-match* method (except for pathological cases not used in practice). The former relies on search all the objects whose similarity or distance to the query satisfies some search constraints (*e.g.* distance to the query less than a threshold). The latter relies on finding an exact copy of the query (*e.g.* distance to the query equal to zero).

[7] The notion of intrinsic dimensionality of a metric space is formally introduced in Section 2.4.2. Intuitively, in spaces where objects are represented as vectors it is the minimum number of coordinates needed to represent the objects without loss of information. Even if general metric objects have not coordinates, it is possible to define a notion of "intrinsic dimensionality" of a metric space that roughly reflects how hard it is to search that space.

efficiency of the approximate search comes at the expense of some reduction of the search accuracy. Successful examples of approximate approach are the *Locality-Sensitive Hashing (LSH)* model [136] within its extensions [46, 174] and the family of *Permutation-Based Indexing (PBI)* [36, 64, 198] approaches.

For a comprehensive introduction on similarity search and metric indexing see for example Chávez et al. [66], Clarkson [69], Hjaltason and Samet [133], and Zezula et al. [271]. In this chapter, we provide mathematical background of metric spaces (Section 2.4.1) and similarity queries (Section 2.4.4). We then review space partitioning (Section 2.4.5) and pruning strategies (Section 2.4.6) that mainly serves as ground for Chapter 5. In Section 2.4.7 we briefly overview some metric access methods before moving on metric space transformations (Section 2.4.8) and permutation-based approximate search (Section 2.4.9).

2.4.1 Metric Space

In the context of similarity search and other applicative domains, the term *distance* is used to indicate a generic function that measure the dissimilarity of two elements. When this function satisfies postulates of non-negativity, identity, symmetry, and triangle inequality, it is called *metric*[8] as defined below:

Definition 2.4.2 (Metric). A *metric* on a set $\mathcal{D}$ is a function $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ meeting the following properties:

$$\forall\, x, y \in \mathcal{D}, \quad d(x, y) \geq 0 \qquad \textit{non-negativity} \tag{2.13}$$

$$\forall\, x, y \in \mathcal{D}, \quad d(x, y) = 0 \quad \text{if, and only if,} \quad x = y \qquad \textit{identity of indiscernibles} \tag{2.14}$$

$$\forall\, x, y \in \mathcal{D}, \quad d(x, y) = d(y, x) \qquad \textit{symmetry} \tag{2.15}$$

$$\forall\, x, y, z \in \mathcal{D}, \quad d(x, z) \leq d(x, y) + d(y, z) \qquad \textit{triangle inequality} \tag{2.16}$$

Definition 2.4.3 (Metric Space). A *metric space* is a pair $(\mathcal{D}, d)$ formed by a set $\mathcal{D}$ and a metric d on it.

There are various way to relax the metric postulates. We say that d is a *pseudometric* if it satisfies all the metric postulates, but instead of the *identity of indiscernibles* only the *reflexivity* condition $d(x, x) = 0$ is required. On the other hand, a function d that satisfies all the metric postulates with the exception of the *symmetry*, it is called *quasimetric*. Note that a quasimetric function can be easily transformed into a metric, for example by considering $d_{\text{sym}}(x, y) = d(x, y) + d(y, x)$. Finally, if the *triangle inequality* does not hold, we talk about *semimetric*.

Some enchantment of the metric postulates can be also considered. For example, a *ultrametric* is a metric also satisfying

$$\forall x, y, z \in \mathcal{D}, \quad d(x, z) \leq \max\{d(x, y), d(y, z)\}$$

which is a property stronger than the triangle inequality. In fact, while the triangle inequality implies that the length of a side of a triangle is always less or equal to the sum of the lengths of other two sides, in the ultrametric space every triangle is isosceles or equilateral. In Chapter 5, we introduce and examine another class of metric spaces, that we call *supermetrics*, which also have geometrical guarantees stronger than the traditional triangle inequality.

In a generic metric space there are no algebraic operations, except for the distance computations. However, most of the spaces that arise in applications are *vectors space*. In this case, the metric objects are vectors that can be added together and multiplied by scalars. When we define metric on vector space we usually require that the metric interact with the algebraic operation. Specifically, suppose that $\mathcal{V}$ is a vector space over $\mathbb{R}$. We usually want to define a metric d on $\mathcal{V}$ such that for each $\mathbf{x}, \mathbf{y} \in \mathcal{V}$, $\lambda \in \mathbb{R}$ the "length" $d(\mathbf{x} - \mathbf{y}, \mathbf{0})$ of the vector $\mathbf{x} - \mathbf{y}$ equals the distance $d(\mathbf{x}, \mathbf{y})$, and the length $d(\lambda\mathbf{x}, \mathbf{0})$ is $|\lambda| d(\mathbf{x}, \mathbf{0})$. In such cases we said that the metric is determined by a length function- the so called *norm*.

[8] It is worth noting that in mathematics, the term "distance" and "metrics" are generally used as synonyms. In the rest of this thesis we use these two terms interchangeably; it will be clear from the context if talking of a metric or not (terms such as "dissimilarity function" or "not-proper metric" are used to avoid ambiguities). Moreover we restrict or analysis on metric space over the field of real numbers $\mathbb{R}$

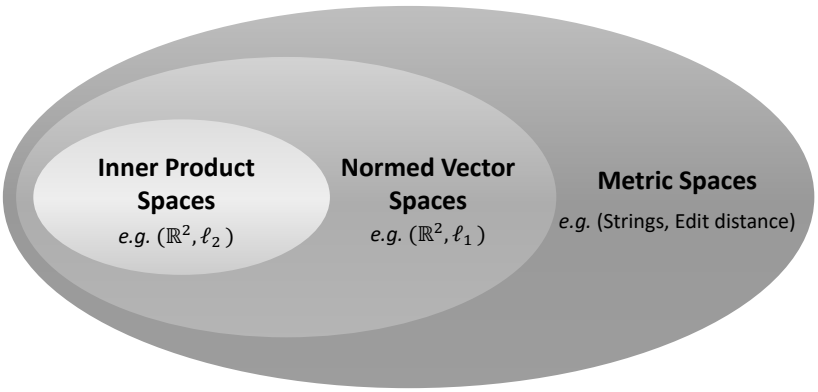


Figure 2.12: *Relations between inner product, norm and metric spaces: an inner product induces a norm, a norm induces a metric. However note that i) not all the metric spaces are metric vector spaces; ii) not all the metric vector spaces are normed vector spaces; iii) and not all the normed vector spaces are inner product spaces.*

Definition 2.4.4 (Norm). A *norm* on a vector space $\mathcal{V}$ is a function $\|\cdot\| : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ such that for each $\mathbf{x}, \mathbf{y} \in V$ and each scalar λ, we have

$$\|\mathbf{x}\| \geq 0 \tag{2.17}$$
$$\|\mathbf{x}\| = 0 \quad \text{if, and only if,} \quad \mathbf{x} = \mathbf{0} \tag{2.18}$$
$$\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\| \tag{2.19}$$
$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|. \tag{2.20}$$

$(\mathcal{V}, ||\cdot||)$ is called *normed vector space* and the number $\|\mathbf{x}\|$ is called *norm* or *length* of $\mathbf{x}$.

Given a normed space it is always possible to define a metric on it:

Proposition 2.4.1. *If $(\mathcal{V}, \|\cdot\|)$ is a normed vector space, then $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ defined by $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ is a metric on $\mathcal{V}$.*

A metric associated with a norm has additional properties with respect to a generic metric function:

$$\forall\, \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{V}, \qquad d(\mathbf{x} + \mathbf{z}, \mathbf{y} + \mathbf{z}) = d(\mathbf{x}, \mathbf{y}) \qquad \textit{translation invariance} \tag{2.21}$$
$$\forall\, \mathbf{x}, \mathbf{y} \in \mathcal{V},\ \forall\, \lambda \in \mathbb{R}, \qquad d(\lambda\mathbf{x}, \lambda\mathbf{y}) = |\lambda| d(\mathbf{x}, \mathbf{y}) \qquad \textit{homogeneity} \tag{2.22}$$

Conversely if $(\mathcal{V}, \|\cdot\|)$ is a vector metric space whose metric d meets the above properties, then $\mathbf{x} \to d(\mathbf{x}, \mathbf{0})$ is a norm. Note that these properties do not make sense in a generic metric space since we cannon add points or multiply them by a scalar.

Finally, we observe that the norm of a vector space (and so the distance) sometimes is induced by an *inner product*; in such cases we tall about *inner product space*.

Definition 2.4.5 (Inner product). An *inner product* on a vector space $\mathcal{V}$ is a function $\langle\cdot,\cdot\rangle : \mathcal{V} \to \mathbb{R}$ such that

$$\forall\, \mathbf{x} \in \mathcal{V} \qquad \langle \mathbf{x}, \mathbf{x} \rangle \geq 0 \tag{2.23}$$
$$\forall\, \mathbf{x} \in \mathcal{V} \qquad \langle \mathbf{x}, \mathbf{x} \rangle = 0 \quad \text{if, and only if,} \quad \mathbf{x} = \mathbf{0} \tag{2.24}$$
$$\forall\, \mathbf{x}, \mathbf{y} \in \mathcal{V} \qquad \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle \tag{2.25}$$
$$\forall\, \mathbf{x}, \mathbf{y}, \in \mathcal{V},\ \alpha \in \mathbb{R} \qquad \langle \alpha\mathbf{x}, \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle \tag{2.26}$$
$$\forall\, \mathbf{x}, \mathbf{y}, \mathbf{z}, \in \mathcal{V} \qquad \langle \mathbf{x} + \mathbf{z}, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{z}, \mathbf{y} \rangle \tag{2.27}$$

Any inner product space is a normed space, and thus a metric space:

Proposition 2.4.2. *If $(\mathcal{V}, \langle\cdot,\cdot\rangle)$ is a inner vector space, then $\|\cdot\| : \mathcal{V} \to \mathbb{R}$ defined by $\|\mathbf{x}\| = \langle \mathbf{x}, \mathbf{x}\rangle$ is a norm on $\mathcal{V}$, and $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ defined by $d(\mathbf{x}, \mathbf{y}) = \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle}$ is a distance on $\mathcal{V}$. Moreover*

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{V} \qquad d(\mathbf{x}, \mathbf{y})^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\langle \mathbf{x}, \mathbf{y}\rangle. \tag{2.28}$$

Figure 2.12 gives simple portrayal of the hierarchy of metric spaces.

2.4.1.1 Distance Measures

In the following, we present some distance functions that are mentioned or used in the remainder of this thesis.

Minkowski Distances (ℓ_p) The family $\{\ell_p\}_{1\leq p\leq\infty}$ is a family of metrics on $\mathbb{R}^n$, called *Minkowski Distances*, defined as

$$\ell_p(\mathbf{x}, \mathbf{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}, \qquad p \geq 1 \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \tag{2.29}$$

$$\ell_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1,\ldots,n} |x_i - y_i|, \qquad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \tag{2.30}$$

The ℓ_1 distance is know as *Manhattan distance* (or *City-Block distance*). The ℓ_2 is the well-known *Euclidean distance*. The ℓ_∞ is called *Chebyshev distance* or *maximum distance*.

The ℓ_∞ is determined by the norm

$$\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|,$$

while for all $1 \leq p < \infty$ the ℓ_p metric is determined by

$$\|\mathbf{x}\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p}.$$

The ℓ_2 is the only of these family of distances that is induced by an inner product, which is the standard dot product $\langle \mathbf{x}, \mathbf{y}\rangle = \sum_{i=1}^{n} x_i y_i$.

In the rest of this thesis we use the notation ℓ_p^n to indicate the metric space $(\mathbb{R}^n, \ell_p)$. For example, $\ell_2^2 = (\mathbb{R}^2, \ell_2)$ is a 2D Euclidean space, $\ell_2^3 = (\mathbb{R}^3, \ell_2)$ is a 3D Euclidean space, etc.

Hamming Distance The *Hamming Distance* [121] is a metric frequently used to compare binary strings. Given two binary strings of equal length, it measures the number of bit positions in which the two strings differ from each other. If we see the two binary string as a point in a real vector space, the Hamming distance equals the ℓ_1 distance.

Note that the Hamming distance can be computed very efficiently with a bitwise XOR operation followed by a bit count.

Cosine Distance (d_{Cos}) The term "cosine distance" does not have a unique meaning in the metric space literature and so requires an explanation.

It has long been known that, for two values $\mathbf{x}, \mathbf{y}$ in $\mathbb{R}^n$, the function

$$S_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}, \tag{2.31}$$

referred to as the *cosine similarity*, gives the cosine of the angle between the vectors, which is a convenient estimate of their dimensional correlation. The cosine similarity is a suitable similarity function in application when the magnitude of the vectors is not important but the main concern is in the direction of

the vectors. One advantage is that this measure is cheap to calculate, especially when the space is sparse such as applications in information retrieval.

As it is bounded in $[0, 1]$, the function $f(\mathbf{x}, \mathbf{y}) = 1 - S_{\cos}(\mathbf{x}, \mathbf{y})$ gives a bounded divergence coefficient; however this function is not a proper metric, as it lacks triangle inequality. A function which gives the same rank order and is also a proper metric can be simply achieved by converting this value into the angle between two vectors, which can be caused to range within $[0, 1]$ by

$$\widetilde{d}_{\text{Cos}}(\mathbf{x}, \mathbf{y}) = cos^{-1}(S_{\cos}(\mathbf{x}, \mathbf{y}))/\pi.$$

In the metric space literature, this metric is sometimes referred to as *Cosine Distance* [75, 101, 235].

However, there exists another rank-equivalent function based on the Cosine similarity:

$$d_{\text{Cos}}(\mathbf{x}, \mathbf{y}) = \sqrt{1 - S_{\cos}(\mathbf{x}, \mathbf{y})}. \tag{2.32}$$

This function is a proper metric and it is equivalent to the Euclidean distance computed on the normalized vectors $\mathbf{x}/\|\mathbf{x}\|$ and $\mathbf{y}/\|\mathbf{y}\|$:

$$d_{\text{Cos}}(\mathbf{x}, \mathbf{y}) = d_{\text{Cos}}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}, \frac{\mathbf{y}}{\|\mathbf{y}\|}\right) = \frac{1}{\sqrt{2}}\left\|\frac{\mathbf{x}}{\|\mathbf{x}\|} - \frac{\mathbf{y}}{\|\mathbf{y}\|}\right\|$$

so it coincides with a rescaled Euclidean distance whenever the vectors are ℓ_2-normalized.

In this thesis, we always use d_{Cos} as *cosine distance*, unless stated specifically otherwise.

Quadratic Form Distance Let $\mathbf{x}$, $\mathbf{y}$ two n-dimensional vectors, and M an $n \times n$ matrix, then the function

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top M(\mathbf{x} - \mathbf{y})} \tag{2.33}$$

is a metric if (and only if) $M = [m_{ij}]$ is a symmetric semi-definite positive matrix [120]. The element m_{ij} denotes how related (or rather, unrelated) the dimensions i and j of the vectors are. When the matrix M is diagonal the corresponding distance is a weighted Euclidean distance.

Jensen-Shannon Distance (d_{JSD}) In information theory, the difference between two probability distributions is often measured by the Kullback-Leibler divergence [156]. The set $M_+^1(A)$ of probability distributions can safely interpret as a set of positive numeric vectors $\mathbf{v} \in \mathbb{R}^n$ for some n where $\sum_i^n v_i = 1$ (although the original definition extends to continuous spaces as well). In such cases, the Kullback-Leibler divergence is computed as

$$KLD(\mathbf{x}, \mathbf{y}) = \sum_i x_i \log \frac{x_i}{y_i}.$$

Note that it is not a metric since both symmetry and triangle inequality postulates are not satisfied. A symmetric variant of KLD is the *Jensen-Shannon divergence*. The term *Jensen-Shannon divergence* is used variously with slightly different meanings; to avoid ambiguity, we define it here as

$$JSD(\mathbf{x}, \mathbf{y}) = 1 - \tfrac{1}{2}\sum_i (h(x_i) + h(y_i) - h(x_i + y_i)), \qquad \mathbf{x}, \mathbf{y} \in M_+^1(A)$$

where

$$h(z) = -z \log_2 z.$$

That formulation, explained in [73], is consistent with other authors and neatly bounds the range into [0,1]. While the Jensen-Shannon divergence does not fulfil the triangle inequality, its squared-root $d_{JSD}(\mathbf{x}, \mathbf{y}) = \sqrt{JSD(\mathbf{x}, \mathbf{y})}$ is a proper metric on $M_+^1(A)$ ([95], [201], [104]).

Triangular Distance (d_Δ) In the space $M_+^1(A)$ of probability distributions it is possible to define the following dissimilarity function

$$\Delta(v, w) = \sum_i \frac{(v_i - w_i)^2}{v_i + w_i}, \qquad x, y \in M_+^1(A) \tag{2.34}$$

which has been named in [248] as *Triangular Discrimination*. It has also occurred in literature with other names, such as "chi-square like distance" [159]. Although rarely used in practice, it is of significant interest as it has relatively tight upper and lower bounds over the much more expensive Jensen-Shannon divergence [248]. The Triangular Discrimination is a semi-metric, but its square root $d_\Delta = \sqrt{(\Delta)}$ is a proper metric on $M_+^1(A)$ [159, 248].

Combination of metrics Finally, we observe that given n metric functions $d_1, \ldots, d_n$ over a domain $\mathcal{D}$, and n non negative value $\alpha_1, \ldots, \alpha_n$, also the combination $d = \sum_{i=1}^n \alpha d_i$ is a metric over $\mathcal{D}$.

2.4.2 Intrinsic Dimensionality

In a vector space, each object is represented as a D-dimensional vector, for some value D called *dimensionality*. It is often the case that an equivalent representation of the vector data can be provided with lower dimensions. For example, suppose to have some vector points in $\mathbb{R}^{10}$ that all lie in a plane. Even if each point is originally described using 10 coordinates, we can represent them using 2-dimensional vectors that are the local coordinates in the plane where they lie. This fact is normally referred in term of *Intrinsic Dimensionality (IDim)* of a set of data that can be lower than the dimensionality of the space where they are defined. In literature, there are several definitions of the intrinsic dimensionality of a vector space. Following Fukunaga [105] and Camastra [62], a dataset contained in $\mathbb{R}^D$ is said to have intrinsic dimensionality equal to M if its elements lie entirely within an M-dimensional subspace of $\mathbb{R}^D$ (where $M < D$). Actually, they differentiate between two possible estimations of the intrinsic dimensionality of a dataset: *local* and *global*. The local one makes the estimation using the information contained in sample neighbourhoods, avoiding the projection of the data onto a lower-dimensional space. The global approach instead uses all the data information to search the minimum M such that the data can be projected in a M-dimensional space without loss of information. Several estimators of the IDim (global or local) for vector spaces have been defined in the literature, [62] gives an excellent survey.

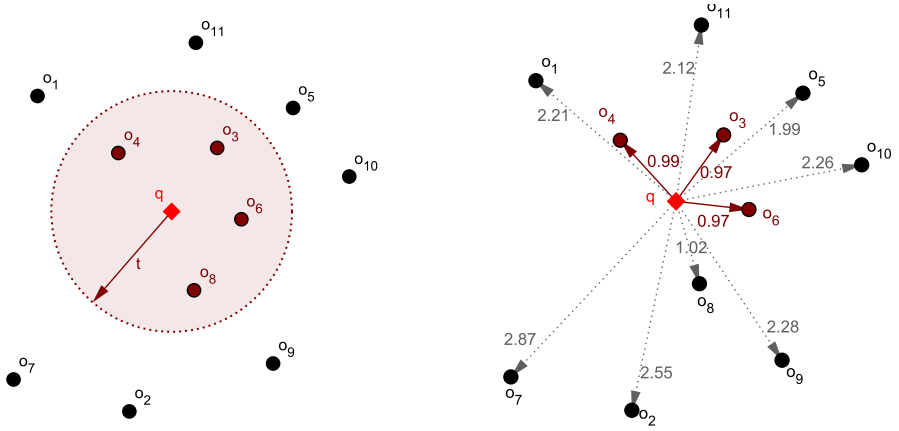
In general metric space the notion of "dimensionality" can be provided in term of the hardness of searching that space using only the distance function to compare the data objects. Chávez et al. [66] proposed to estimate the IDim of a metric space as $\mu^2/(2\sigma^2)$ where μ is the mean and σ is the standard deviation of the histogram of distances between points in the metric space. The intuition at the core of this definition is that in random D-dimensional vector spaces the histogram of the distances has a larger mean and smaller variance as the D increase. So if the histogram of distances for a set of metric data has large mean and small variance is liked to have high intrinsic dimensionality. This definition is very popular to estimate the IDim of a metric space, mainly because it is simple to calculate from sampling pairs of the space and gives a reasonable measure of "intrinsic search difficulty" of the space. We used this measure in the later chapters. It is worth mentioning that several different metric IDim estimators have been proposed in the literature, including fractal-based methods, correlation-based methods, Distance exponents, PCA-based methods. We refer to [192] for further details.

2.4.3 Efficiency Measures for Exact Search

The cost to answer a query are highly influenced the partitioning principle used to organize the data and the query execution algorithm. As argued by Chávez and Navarro [66], the total cost can be split as

$$C = \#\text{ distance evaluations} \times \text{complexity of } d(\cdot) + \text{extra CPU time} + \text{I/O time}.$$

In many cases, only the number of distance evaluations is used as measure of the complexity of a search algorithm, given that (i) in many applications the cost of the distance computations is so high that the



$R(q, 1.5) = \{o_3, o_4, o_6, o_8\}$

(a) *Range query*

$k - NN(q) = \{o_3, o_6, o_4\}$

(b) *k-Nearest Neighbour Query (k=3)*

Figure 2.13: *Examples of a range query (a) and a k-NN query (b).*

other components of the cost can be neglected, (ii) in general, the raw timing and the extra CPU cost can vary quickly as hardware evolves, and (iii) I/O time can be the dominant factor in some applications and negligible in others. Clearly, for a dataset of N objects queries can be answered by evaluating N distances (sequential scan). The main goal is organize the data such that each query can be answered with a sublinear cost in term of distances evaluations. However, in some cases counting only the number of distances computations is not a good measure of the overall efficiency, as for example when evaluating new indexing mechanisms.

2.4.4 Similarity Queries

In real applications, we are interested in searching a (large) finite set of objects $\mathcal{S}$ which is a subset of a data domain $\mathcal{D}$. The general requirement is to efficiently find members of $\mathcal{S}$ which are similar to an arbitrary member of $\mathcal{D}$ (the *query object*), where the distance function d gives the only way by which any two objects may be compared - the bigger the distance $d(x, y)$, the less similar the data $x, y \in \mathcal{D}$. Notice that the query is a generic object of the space $\mathcal{D}$ and so it may not exist in the collection $\mathcal{S} \subset \mathcal{D}$ to be searched.

A *similarity query* is defined by a query object and a proximity condition; the response to the query is the set of the objects satisfying the considered condition with respect to the query. The most common types of similarity queries are the *range query* and *k-nearest neighbour query*. For a comprehensive description of similarity queries see [271, ch.4].

2.4.4.1 Range Query

Probably the most simple type of similarity query is the *range* search query $R(q, t)$, which is specified by a query object $q \in \mathcal{D}$ and threshold distance $t \in \mathbb{R}$. It retrieves all the objects found within distance t of q, *i.e.*

$$R(q, t) = \{x \in \mathcal{S} | \, d(x, q) \leq t\}.$$

Figure 2.13a shows an example of range query in a planar domain.

The basic strategy to answer a range query $R(q, t)$ is to examine all the database objects, compute the distance of the objects to the query, and report on the result set only those within distance t from q. This general algorithm is adapted and enchanted by further filtering and pruning strategies on the basis of the index structure used to organize the data objects. Some of them are described in the Sections 2.4.6 and 2.4.7.

2.4.4.2 Nearest Neighbour Query

Given a query object q and a integer k, the *k-Nearest Neighbour* query $kNN(q)$ finds the k objects in $\mathcal{S}$ closest to q (Figure 2.13b). Formally, $k - NN(q)$ is a subset of $\mathcal{S}$ such that

$$|k - NN(q)| = k \tag{2.35}$$

$$\forall\, x \in k - NN(q), \forall\, y \in \mathcal{S} \setminus k - NN(q) \qquad d(x,q) \leq d(y,q) \tag{2.36}$$

If several objects lie at the same distance from the k-th nearest neighbour, the ties are solved arbitrarily.

The k-nearest neighbour is often adopted as search paradigm since, as highlighted in [96] and [203], it allows us to control the size of the results set, and it is simpler to be used in high-dimensional space where it is not obvious to define a meaningful distance value to be used with other search paradigms such as the range query.

The basic strategy for evaluating a k-NN query (without assuming any index structure of the data) incrementally builds the result set as follows: Initially select k objects and order them with respect to the distance from q. Let t_k the distance of the k-nearest object in the result set. All the other objects o_i are consecutively scanned and if the distance $d(o_i, q)$ is smaller t_k the object o_i is inserted in the response and previous k-th nearest neighbour is removed. The threshold t_k is updated to $d(o_i, q)$ and the process continue. To achieve sub-linear search complexity, index structures and pruning rules are typically used.

2.4.5 Space Partitioning

Typically the search space $\mathcal{S}$ is too large to allow an exhaustive search (*i.e.* examining the entire data set). In major cases, the data can be preprocessed and partitioned in order to build an index data structure that allow saving distance computations when answering a query.

Each index is constructed off-line and it is maintained by a specificity designed access method. Metric access methods make use of properties of the metric governing the space to arrange the data objects in such a way as to minimize the time required to retrieve the query results. At query time, properties of the metric space (*e.g.* triangle inequality) are exploited to determine subsets of $\mathcal{S}$ that do not need to be exhaustively checked. Such avoidance is normally referred to as *exclusion* or *space pruning* (several pruning strategies are described later in Section 2.4.6).

2.4.5.1 Ball Partitioning

Ball partitioning [252] entails the selection of a reference object $p \in \mathcal{D}$, a covering radius $r \in \mathbb{R}$, and the subdivision of the object in the search space $\mathcal{S}$ based on their distance from p. Formally $\mathcal{S}$ is partitioned into two subsets:

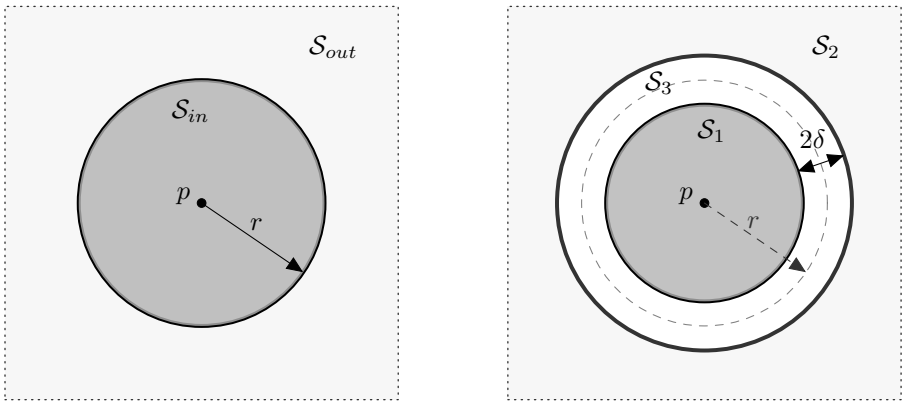
$$\mathcal{S}_{in} = \{x \in \mathcal{S} \,|\, d(x,p) \leq r\} \tag{2.37}$$

$$\mathcal{S}_{out} = \{x \in \mathcal{S} \,|\, d(x,p) > r\}. \tag{2.38}$$

See Fig. 2.14 for example. Typically, the covering radius r is chosen to be equal to the median of $\{d(x,p),\, \forall x \in \mathcal{S}\}$ in order to have a balanced split. In that case, the redundant condition $d(x,p) \geq r$ is usually added in the definition of S_{out} so that each element at the median distance can be assigned to one of the subsets in an arbitrary, but balanced, fashion [271, ch.5]. Some of the most important techniques using ball partitioning are: Burkhard-Keller Tree (BKT) [59], Fixed Queries Tree (FQT) [44], Vantage Point Tree (VPT) [266]. We describe VPT in Section 2.4.7.1.

2.4.5.2 Excluded-Middle partitioning

The *Excluded-middle partitioning* [267] is another example of radius-based partitioning (see Fig. 2.14b). We do not explicitly use this partitioning principle in the thesis, but we report it for reference. This technique divides the search space into three subsets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$, on the basis of a covering radius r and a



(a) *Ball Partitioning* **(b)** *Excluded-Middle partitioning*

Figure 2.14: *Examples of Radius-Based Partitioning.*

threshold δ:

$$\mathcal{S}_1 = \{x \in \mathcal{S} \,|\, d(x,p) \leq r - \delta\} \tag{2.39}$$
$$\mathcal{S}_2 = \{x \in \mathcal{S} \,|\, d(x,p) > r + \delta\} \tag{2.40}$$
$$\mathcal{S}_3 = \{x \in \mathcal{S} \,|\, r - \delta < d(x,p) \leq r + \delta\}. \tag{2.41}$$

As pointed by Zezula et al. [271, ch.5], the excluded-middle partitioning can be envisaged as an extension of the ball partitioning principle, motivated by the fact that when a query lies near the border of the ball $\mathcal{S}_{in}$ the search process requires to access to both the partitions $\mathcal{S}_{in}$ and $\mathcal{S}_{out}$. The central idea of the excluded-middle partitioning is to form a third partition (of thickness 2δ), containing points near the border of $\mathcal{S}_{in}$, so that when considering a range search $R(q,t)$ with $t < \delta$, for sure we can exclude at least one region.

Two examples of index structure built using this partitioning principle are the Excluded Middle Vantage Point Forest [267] and the D-Index [93]

2.4.5.3 Generalized Hyperplane Partitioning

The Generalized Hyperplane Partitioning [252] breaks the set $\mathcal{S}$ into two subsets according to the distances of data objects to two reference objects p_1, p_2.Formally:

$$\mathcal{S}_1 = \{x \in \mathcal{S} \,|\, d(x,p_1) \leq d(x,p_2)\} \tag{2.42}$$
$$\mathcal{S}_2 = \{x \in \mathcal{S} \,|\, d(x,p_1) \geq d(x,p_2)\} \tag{2.43}$$

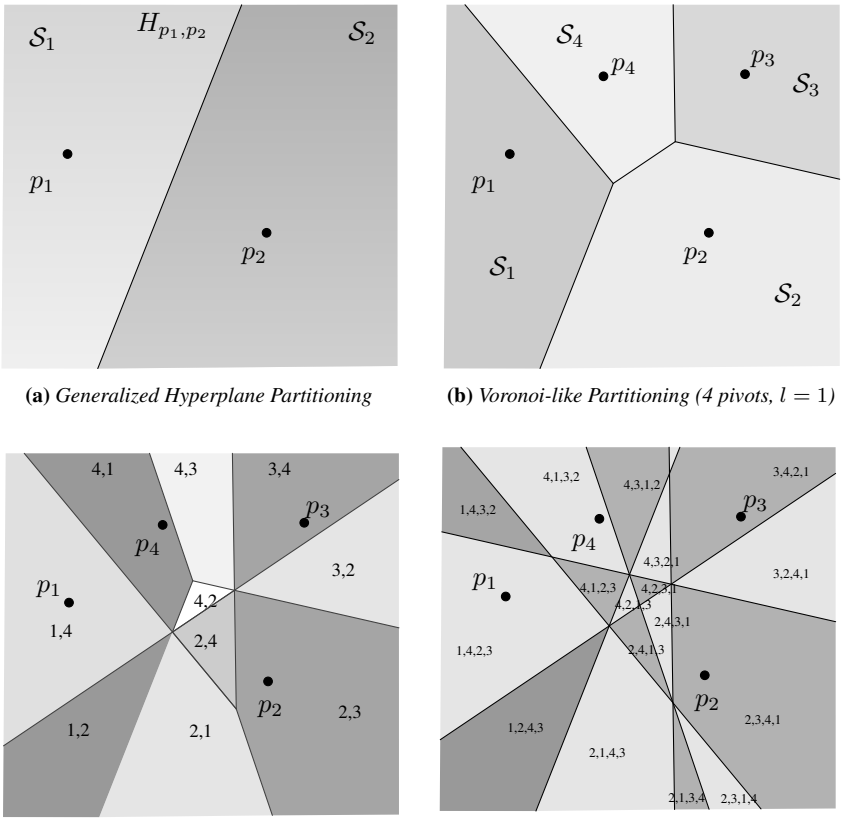
Objects with the same distance to both p_1 and p_2 can be assigned to either $\mathcal{S}_1$ or $\mathcal{S}_2$. An example of hyperplane partitioning is given in Fig. 2.15a. In general, the hyperplane partitioning does not grantee a balances split, and a suitable choice of pivots to achieve this is challenging in generic metric space.

Some of the most important indexing techniques using generalized hyperplane partitioning are: Generalized Hyperplane Tree (GHT) [252], Bisector Tree (BST) [149], and Monotonous Bisector Tree (MBT) [197], which we describe in Sections 2.4.7.2 and 2.4.7.3.

2.4.5.4 Voronoi-like partitioning

An extension of generalized hyperplane partitioning is the *Voronoi* (or *Dirichlet*) *partitioning* [42]. In this type of partitioning, a finite set of pivot points $p_1, \ldots, p_n \in \mathcal{D}$ are chosen, and the rest of $\mathcal{S}$ is divided into n subsets according to which of these pivots is closer. So Voronoi-like partitioning it can be viewed as a "*multiway generalized hyperplane partitioning*" [127, pag.212]. Formally:

$$\mathcal{S}_i = \{x \in \mathcal{S} \,|\, d(x,p_i) \leq d(x,p_j) \quad \text{for all } j \neq i\} \qquad i = 1, \ldots n. \tag{2.44}$$



(a) *Generalized Hyperplane Partitioning*

(b) *Voronoi-like Partitioning (4 pivots, $l = 1$)*

(c) *Voronoi-like Partitioning (4 pivots, $l = 2$)*

(d) *Voronoi-like Partitioning (4 pivots, $l = 4$)*

Figure 2.15: *Examples of Hyperplane-Based Partitioning.*

An example of Voronoi-like partition is depicted in Fig. 2.15b.

The Voronoi partitioning can be applied recursively. The first level ($l = 1$) is the one described above, where each object is assigned to the closest pivot. On the second level ($l = 2$) each cluster S_i is partitioned into at most $n - 1$ subsets $\mathcal{S}_{i,j}$ using the $n - 1$ pivots $\{p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n\}$, where $\mathcal{S}_{i,j} = \{x \in \mathcal{S}_i \,|\, d(x, p_j) \leq d(x, p_k) \quad \text{for all } k \neq i, k \neq j\}$. This partitioning process can be repeated l time, with $l \leq n$. Fig. 2.15c and Fig. 2.15d show an example of Voronoi-like partitioning for respectively level $l = 2$ and $l = 4$.

Please note that the Voronoi partitioning are uniquely determined by the hyperplanes $H_{p_i,p_j} = \{x \in D \,|\, d(x, p_i) = d(x, p_j)\}$ and their intersections. In theory, at level l we have a maximum of $n(n-1)\cdots(n-l+1)$ partitions; however, not all of these partitions really exist for a particular space and set of pivots due to geometrical constrains [234]. For example, in the case depicted in Fig. 2.15c the cluster $\mathcal{S}_{1,3}$ and $\mathcal{S}_{3,1}$ do not exist.

Voronoi-like partitioning of metric space is at the core of Geometric Near-neighbor Access Tree (GNAT) [57], Metric Index (M-index) [198], and other PBI techniques [36, 96]. We discuss PBI techniques in Section 2.4.9.2.

2.4.6 Pruning Strategies

Pruning conditions allow excluding certain regions or objects from the searching process. As observed by Zezula et al. [271, ch.7], "*pruning conditions must be applied not only to avoid accessing irrelevant sets of objects, but also to minimize the number of distances computed.*" The main rationale behind such strategies is to use already-evaluated distances between data objects and some reference objects, and exploit metric postulates (in particular, the triangle inequality) to determine bounds on distances

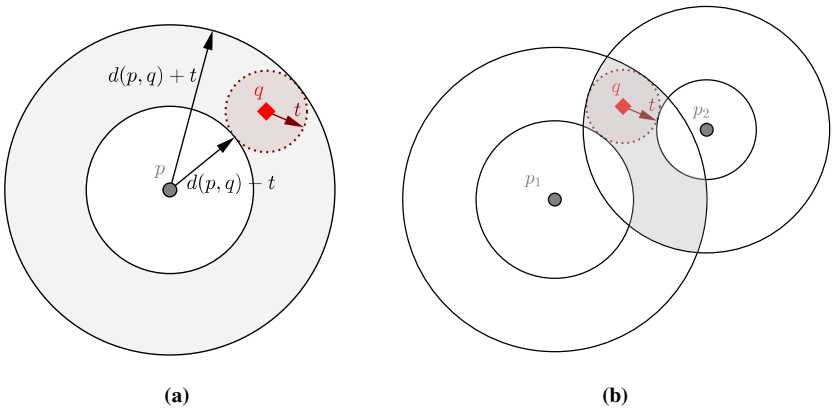


Figure 2.16: *Illustration, in a* planar *domain, of the Object-Pivot pruning rule (a) and its generalization using two pivots (b). Any solution to the query $R(q,t)$ must lie in the regions bounded by the arcs shown in the figure.*

between the query and the other data objects.

For *exact* metric search, almost all indexing methods can be divided into those which use a single reference point to give radius-based exclusion, and those which use two or more reference points to give hyperplane-based exclusion. Many variants of each have been proposed, including many hybrids; [271], [66], [225] [133] give excellent surveys.

In the following, we report several bounding strategies and related pruning rules that are employed, in a specific form, in practically all metric access methods. In Chapter 5 we will define a new pruning rule, which we named *Hilbert Exclusion*, that allows any indexing mechanism which uses hyperplane partitioning to perform better in many common metric spaces.

2.4.6.1 Object-Pivot Distance Constraint

A fundamental distance bounding condition is the *object-pivot distance constraint*, which is a direct consequence of the triangle inequality.

Lemma 2.4.1 (Object-Pivot Distance Constraint [271])**.** *Given a metric space $(\mathcal{D}, d)$, and three arbitrary objects $q, p, o \in \mathcal{D}$, it always guaranteed:*

$$|d(p,q) - d(p,o)| \leq d(q,o) \leq d(p,q) + d(p,o). \tag{2.45}$$

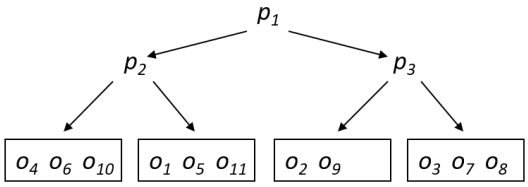
Consequently, the distance $d(q,o)$ can be bounded from below and above, provided the distance $d(p,q)$ and $d(p,o)$ are know.

Corollary. *Given a rage query $R(q,t)$, an object o and a pivot p. Let $m = d(o,p)$ a precomputed distance; then*

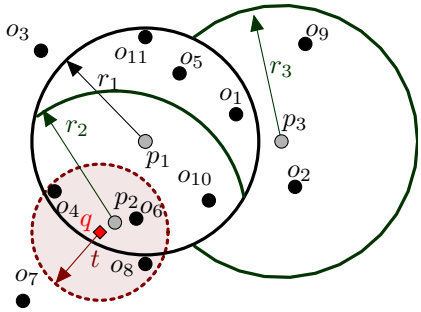
$$\begin{aligned} d(p,q) \leq t - m \quad &\Rightarrow \quad d(q,o) \leq t \quad \textit{[o is a solution]} \\ |d(p,q) - m| > t \quad &\Rightarrow \quad d(q,o) > t \quad \textit{[o can be excluded]} \end{aligned} \tag{2.46}$$

This pruning rule is typically used when the dataset is partitioned according to the distances of the data objects from one or more pivots. Such distances are stored during the insertion of the objects in the data structure so that at query time some object o can be pruned or selected as a solution without explicitly compute $d(q,o)$ (see Figure 2.16).

Fig 2.17 shows an example, reported from [271, p.27], in which this pruning rule is beneficial with respect to the sequential scan of all the data objects. Suppose that an index structure is built over eleven objects o_i using a ball-partitioning principle in a recursively manner. The first level of the tree is obtained



(a) *Binary tree built using a recursive ball partitioning of the space* $\mathcal{S} = \{o_1, \ldots, o_{11}\}$



(b) *Geometric point of view (ball partitioning of the space and range search* $R(q,t)$ *for* $t = 1.5$*)*

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}	o_{11}
p_1	2.1	3.5	3.0	2.5	1.7	1.7	4.4	2.7	4.1	1.9	2.3
p_2	3.6	4.1	-	1.6	3.6	0.7	-	-	-	2.1	4.1
p_3	-	-	5.7	-	-	-	6.7	4.2	2.2	-	-

(c) *Precomputed Object-Pivot distances (computed at building time)*

$$d(q, p_1) = 2.2$$
$$d(q, p_2) = 0.4$$
$$d(q, p_3) = 4.5$$

(d) *Query-Pivot distances*

Figure 2.17: *Example of an index structure built over eleven objects* $\{o_1, \ldots, o_{11}\}$ *obtained using recursively a ball-partitioning principle (example and visualization adapted from [271]).*

using the pivot p_1: all the object inside the ball of centre p_1 and radius r_1 are stored in the lefty sub-tree, the others are stored in the right sub-tree. The sub-trees are recursively partitioned using the same principle with the pivots p_2 and p_3. In general, given this binary tree, a range search $R(q,t)$ is performed in a top-down fashion: it starts at the root node and decides which sub-trees must be visited. The pruning rules helps to optimize the search procedure. In particular, the object-pivot distance constrain is usually applied to leaf nodes containing the data. For example, assume that a range query $R(q,t)$ with $t = 1.5$ is issued and that the search algorithm has reached the left-most leaf node containing objects $\{o_4, o_6, o_{10}\}$. At this stage, instead of examine all the object of the leaf, which would require three distance computations, we can compute only the distance between the query and the pivot p_2 and exploit the precomputed distances $d(p_2, o_4)$, $d(p_2, o_6)$ and $d(p_2, o_{10})$ to omit some query-object distance evaluations. Since the absolute value of the difference between $d(p_2, o_{10})$ and $d(p_2, q)$ is greater than the query threshold t, o_{10} can be excluded without computing the actual distance $d(q, o_{10})$. Moreover, since $d(p_2, q) + d(p_2, o_6)$ is smaller that the query threshold, the object o_6 can be directly included in the solution set, saving one more distance computation. For what concern the object o_4, since $d(p_2, q) + d(p_2, o_4) > t$ and $|d(p_2, q) - d(p_2, o_4)| < t$ we cannot either include or exclude the point from the solution set without computing the actual distance $d(q, o_4)$.

If the distance of an object is stored for multiple pivots $\{p_1, \ldots, p_n\}$, the previous pruning rule can be applied for each pivot an so the distance between $d(o, q)$ can be avoided if

$$\max_i |d(q, p_i) - d(p_i, o)| > t. \tag{2.47}$$

This condition, referred to as *pivot filtering* [92], is a direct consequence of the following generalization of the Lemma 2.4.1:

Lemma 2.4.2. *Given a metric space* $(\mathcal{D}, d)$*, a set of* n *pivots* p_i *and two arbitrary objects* $q, o \in \mathcal{D}$*, it always guaranteed:*

$$\max_{i=1,\ldots,n} |d(p_i, q) - d(p_i, o)| \leq d(q, o) \leq \min_{i=1,\ldots,n} (d(p_i, q) + d(p_i, o)). \tag{2.48}$$

Consequently, the distance $d(q, o)$ can be bounded from below and above, whenever the distances $d(p_i, q)$ and $d(p_i, o)$ are know.

The filtering power of pivot-based exclusion can be augmented by selecting good set of pivots, as proposed in [58, 60].

2.4.6.2 Range-Pivot Distance Constraint

Some index structures do not store all distances between database objects and pivots. In order to minimize the space needed to build the index, an alternative is to store only the minimum and the maximum distance of the objects from the pivots (*covering radii*). In such case, a weaker form of the object-pivot distance constraint, called *range-pivot distance constraint*, is applied.

Lemma 2.4.3 (Range-Pivot Distance Constraint [271])**.** *Given a metric space $(\mathcal{D}, d)$ and objects $p, o \in \mathcal{D}$ such that $r^{min} \leq d(p, o) \leq r^{max}$ with $r^{min}, r^{max} \in \mathbb{R}^+$, and given some $q \in \mathcal{D}$, the distance $d(q, o)$ is bounded as follow:*

$$\max\{d(q,p) - r^{max}, r^{min} - d(q,p), 0\} \leq d(q,o) \leq d(q,p) + r^{max}. \tag{2.49}$$

Corollary. *Given a range query $R(q, t)$ and a cluster $\mathcal{S}_p$ such that for all $x \in \mathcal{S}_p$ we have $r^{min} \leq d(p, o) \leq r^{max}$, then*

$$\begin{aligned} d(q,p) \leq t - r^{max} \quad &\Rightarrow \quad \textit{All the objects in } \mathcal{S}_p \textit{ are solutions} \\ \max\{d(q,p) - r^{max}, r^{min} - d(q,p)\} > t \quad &\Rightarrow \quad \mathcal{S}_p \textit{ can be excluded} \end{aligned} \tag{2.50}$$

2.4.6.3 Pivot-Pivot Distance Constraint

Whenever the distance $d(q, p)$ is not explicitly computed, but a high and lower bounds of it are provided, we can use the following principle:

Lemma 2.4.4 (Pivot-Pivot Distance Constraint [271])**.** *Given a metric space $(\mathcal{D}, d)$ and objects $q, p, o \in \mathcal{D}$ such that $r^{min} \leq d(p, o) \leq r^{max}$ and $r_q^{min} \leq d(p, q) \leq r_q^{max}$, the distance $d(q, o)$ is bounded by the range:*

$$\max\{r_q^{min} - r^{max}, r^{min} - r_q^{max}, 0\} \leq d(q,o) \leq r_q^{max} + r^{max}. \tag{2.51}$$

Corollary. *Given a range query $R(q, t)$, a pivot p, and a cluster $\mathcal{S}_p$ such that for all $x \in \mathcal{S}_p$ we have $r^{min} \leq d(x, p) \leq r^{max}$. If $r_q^{min} \leq d(p, o) \leq r_q^{max}$ then*

$$\begin{aligned} r_q^{max} \leq t - r^{max} \quad &\Rightarrow \quad \textit{All the objects in } \mathcal{S}_p \textit{ are solutions} \\ \max\{r_q^{min} - r^{max}, r^{min} - r_q^{max}\} > t \quad &\Rightarrow \quad \mathcal{S}_p \textit{ can be excluded} \end{aligned} \tag{2.52}$$

2.4.6.4 Double-Pivot Distance Constraint

Differently from the previous bounding rules, the *double-pivots distance constrain*, is based upon generalized hyperplane partitioning. Let p_1 and p_2 two pivots used to partition the space $\mathcal{S}$ into

$$\begin{aligned} \mathcal{S}_1 &= \{x \in \mathcal{S} \,|\, d(x,p_1) \leq d(x,p_2)\} \\ \mathcal{S}_2 &= \{x \in \mathcal{S} \,|\, d(x,p_1) \geq d(x,p_2)\}. \end{aligned}$$

According to the triangle inequity and the Lemma 2.4.1 (object-pivot distance constraint), any solution s to the query $R(q, t)$ is such that

$$d(q,p_i) - t \leq d(s,p_i) \leq d(q,p_i) + t \qquad \text{for} \quad i = 1, 2.$$

Assume, without loss of generality, that the query q lies in $\mathcal{S}_2$, then if $d(q, p_1) - d(q, p_2) > 2t$ we have $d(s, p_1) - d(s, p_2) \geq 0$, which implies that $s \in \mathcal{S}_2$ and so $\mathcal{S}_1$ can be excluded.

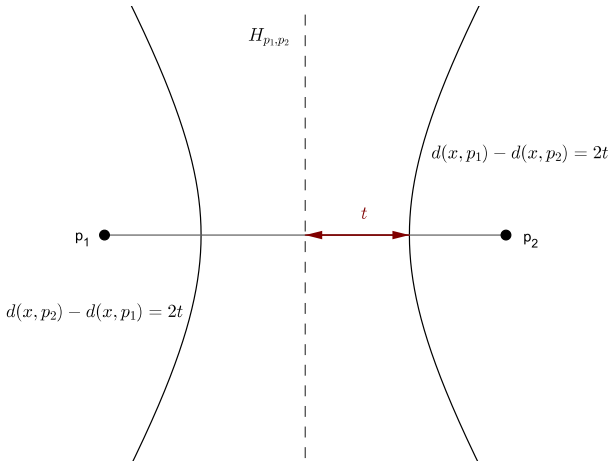


Figure 2.18: *Assume to have a generalized Hyperplane partitioning of the space given two pivots p_1 and p_2. The hyperplane H_{p_1,p_2} divided the space into two subsets according to which of the two pivots is closer. The boundary defined by the exclusion condition $|d(q,p_1) - d(q,p_2)|/2 > t$ is a hyperbola focused at p_1 and p_2, with semimajor axis t.*

Lemma 2.4.5 (Double-Pivot Distance Constraint [271])**.** *Assume a metric space $(\mathcal{D}, d)$, the objects $p_1, p_2 \in \mathcal{D}$, and on object $o \in \mathcal{S}_1$. Given a query object $q \in D$ and the distances $d(q,p_1)$ and $d(q,p_2)$, the distance $d(q,o)$ is lower-bounded as follows:*

$$\max\{d(q,p_1) - d(q,p_2)/2, 0\} \leq d(q,o). \tag{2.53}$$

Corollary (Hyperbolic Exclusion)**.** *Assume a range query $R(q,t)$ and a hyperplane partitioning of the space $\mathcal{S}$ for the two pivots $p_1, p_2 \in D$. If $q \in \mathcal{S}_1$ then*

$$\frac{d(q,p_2) - d(q,p_1)}{2} > t \quad \Rightarrow \quad \mathcal{S}_2 \text{ can be excluded.} \tag{2.54}$$

By symmetry, if $q \in \mathcal{S}_2$ then

$$\frac{d(q,p_1) - d(q,p_2)}{2} > t \quad \Rightarrow \quad \mathcal{S}_1 \text{ can be excluded.} \tag{2.55}$$

Regarding the example showed in Figure 2.16, since the query is closer to the pivot p_2 than p_3 and since $(d(q,p_3) - d(q,p_2))/2 = 2.05$ is bigger than $t = 1.5$ we can exclude the left subtree (objects o_2, o_3, o_7, o_8, o_9) saving 6 distance computations. Please, also note that this constrain does not employ any already evaluated distance from a pivot to a database object, but only the distances between pivots and the query.

We refer to the pruning rule 2.54 as *Hyperbolic Exclusion* to recall that all possible positions of the query object q with a constant value $(d(q,p_2) - d(q,p_1))/2$ form a hyperbolic curve, which is the boundary of the exclusion condition (see Figure 2.18).

2.4.7 Metric Access Methods for Exact Search

In this section, we present some basic index structures that exploit the partition and pruning rules introduced in the previous section. We focus only on structures that are used or referred in the remainder of this thesis. More comprehensive surveys can be found in literature [66, 133, 271].

2.4.7.1 Vantage Point Tree

The *Vantage Point Tree (VPT)* [266] recursively partition the space using the ball partitioning principle with median distance as covering radius. This lead to the construction of a balanced binary tree. Starting with the whole dataset, this technique select a pivot p (also called *vantage point*) and split the data into two subsets on the basis of the median distance r_m from the pivot. So data points with distance from p less than or equal to r_m are stored in the left subtree, and the points with a distance greater than or equal to r_m are stored in the right subtree. Each subtree is then recursively partitioned by selecting a pivot and using the same ball partitioning principle. One or more objects can be stored in the leaves.

The search algorithm for a range query $R(q, t)$ traverses the VPT from the root to the leaves. For each internal node, the distance between the actual pivot and the query is evaluated to decide if include or not the pivot in the result set. Then the distances between the query and the pivots of the left and right subtree are evaluated in order to decide which subtree needs to be accessed according to the pruning rules reported in the Corollaries of Lemma 2.4.3 (range-pivot distance constraint) and Lemma 2.4.1 (object-pivot distance constraint). Note that both subtrees can be visited simultaneously.

2.4.7.2 Generalized Hyperplane Tree

The *Generalized Hyperplane Tree (GHT)* [252] recursively partition a dataset using the hyperplane partitioning: Two pivots p_1 and p_2 are selected at each node, the objects closest to p_1 are stored in the left subtree, the ones closest to p_2 are stored in the right subtree. In this way a binary tree (not necessarily balanced) is build over the data, such as the example showed in Figure 2.16.

The range search algorithm, in every step of the traversal, typically uses the Hyperbolic exclusion (Corollary of Lemma 2.4.5) to decide which subtree to access.

2.4.7.3 Bisector Tree and Monotone Bisector Tree

The *Bisector Tree (BST)* [149], similarly to GHT, recursively divides the space using the generalized hyperplane partitioning. Differently from GHT, BST stores at each node the covering radii, *i.e.* the maximum and the minimum distance between the pivot and any objects in its subtree. The information about the covering radii allows exploiting the radius-pivot distance constraint (Lemma 2.4.3) for space pruning during a range search.

A variant of BST is the *Monotonous Bisector Tree (MBT)* [197]. The main idea of MBT is re-using pivots of the internal nodes so that fewer distance computations are needed to build the index and execute a query. Specifically, pivots representing the left and the right subtrees are copied to the corresponding left and right child internal nodes.

2.4.7.4 Spatial Approximation Trees

The *Spatial Approximation Tree (SAT)* [191], also referred to as *sa-tree*, is a data structure that approaches to the searched object spatially: it starts at some point in the space and gets iteratively closer to query. Its core idea is building a graph where nodes are elements of the search space $\mathcal{S}$ and edges connect each object (node) to its "strictly" neighbours. Having such ideal data structure allows one to spatially traverse the search space by visiting neighbour objects that are closer and closer to a given database object. However, this graph cannot uniquely be determined in a generic metric space [191, 271]. A simplified structure is built instead: A root node p is arbitrarily chosen. Then a minimal set $N(p)$ of its neighbours are selected such that

$$o \in N(p) \quad \Longleftrightarrow \quad \forall x \in N(p) \setminus \{o\}, \quad d(o,p) < d(o,x).$$

This means that $N(p)$ contains all the objects that are closer to p than to any other element of $N(p)$. Computing $N(p)$ is a non-trivial optimization problem so some heuristics (see [191]) are used in practice. Once computed $N(p)$ all the other elements not in $N(p) \cup \{p\}$ are assigned to their closest element in $N(p)$. Each element in $N(p)$ is then recursively used as the root of a new subtree. The nodes of the tree store their covering radii, *i.e.* the maximum distance between a node and any elements of the subtree rooted at it. During the search SAT uses both Hyperbolic exclusion and pruning rules based on covering radius: for a range query $R(q, t)$, and a given node a with covering radius $r_a^{\max}$ then

1. if $d(q,p) - r_a^{\max} > t$ then there is no need to consider the subtree rooted at a;
2. let c the closest element to q between $\{a\} \cup N(a) \cup \{\text{ancestors of } a\}$, for any $b \in N(a)$ if $d(q,b) - d(q,c) > 2t$ there is no need to visit the children b.

It is worth noting that the tree is not balanced and its shape depends on the chosen root node (different selections of the root will generate different trees with likely different search costs). Moreover, SAT is a static data structure: once built it cannot handle object insertions or deletion.

A dynamic version of SAT, the *Dynamic Spatial Approximation Tree (DSAT)*, was proposed in [193]. The DSAT is built incrementally, via insertion, but in this case the arity of the nodes is fixed and the neighbours to a node are selected in the first-come-first-serve basis. As a consequence of this building mechanism, a parent node is always "older" than its children (according to the insertion timestamps).

As highlighted by Chávez et al [65]: "*A very surprising and unintended feature of the DSAT is the boosting in the searching performance*". In fact, it was experimentally proved that SAT it is slower than its dynamic version (for certain arity combinations), even if at construction time the former has a full knowledge of the search space [194]. This unexpected result was further investigated in [65], where it was showed that changing only the insertion policy of SAT and leaving all the other settings unchanged highly boost the performance. They proved that even a random insertion policy produce a faster version of SAT. Moreover they proposed to select "distal" instead of "proximal" nodes, which actually is the opposing of the original SAT insertion policy. The resulting tree, named *Distal Spatial Approximation Tree (DiSAT)* [65], showed promising results. The main rationale for using distal nodes is it increases the separation of the hyperplanes while reducing the size of the covering radius, which allows more exclusions at query time.

2.4.7.5 AESA and LAESA

The *Approximating and Eliminating Search Algorithm (AESA)* [258] is an indexing techniques based on pre-computing all the object-to-object distances, which are then exploited to give efficient answers to similarity search queries.

For a dataset of m objects $\{o_1, \ldots, o_m\}$, AESA computes the $m(m-1)/2$ distances between all pairs of objects. The distances are stored in a $m \times m$ symmetric matrix (or equivalently a table): the element at position ij is the distance $d(o_i, o_j)$.

During the search for a range query $R(q,t)$, some data objects are used as pivots to filter and refine a set of candidate results. The initial candidate set contains all the data objects. An object p is first picked at random and used as a pivot: the distance $d(q,p)$ is computed and used for pruning some object according to the object-pivot distance constraint (Lemma 2.4.1). Specifically every object o such that $|d(p,q) - d(p,o)| < t$ is eliminated from the candidate set. Note that $d(p,o)$ is a pre-computed distance. The search algorithm then chooses another pivot from the remaining objects and repeats the previous procedure to further eliminate some data objects. This process is repeated until the size of the candidate set is small enough; then each candidate object o is compared with the query (*i.e.* the actual distance $d(o,q)$ is evaluated) and objects satisfying $d(o,q) \leq t$ are returned in the result set.

One main drawback of AESA is that both space and construction time complexity is quadratic with the number m of data objects. For this reason AESA is applicable when indexing rather small data sets.

The *Linear AESA (LAESA)* [184] reduces this complexity by storing distances from objects to only a fixed number n of pivots. The search procedure is nearly the same of AESA, except that not all the objects are used as pivots, and that the pivot filtering (Eq.(2.47)) is used to filter out the objects. As for AESA, objects in the resulting candidate set are then directly compared to the query object.

2.4.8 Metric Space Transformations

It is sometimes the case that a metric space transformation is applied before indexing a dataset or answering to a query. The central idea is to project all the metric objects and the query itself to a new metric space where the results of a query (or candidate set for it) can be efficiently computed. Typically in the projected space the metric is less expensive then the original metric or less distance computation are required to answer a query.

For example the searching process used in LAESA (Section 2.4.7.5) is based on the following *metric embedding*:

$$\begin{aligned}\Phi : (\mathcal{D}, d) &\rightarrow (\mathbb{R}^n, \ell_\infty) \\ o &\rightarrow [d(o, p_1), \ldots, d(o, p_n)]\end{aligned}$$

given the set of pivots $\{p_1, \ldots, p_n\}$. Then for a range query $R(q, t)$, if $\ell_\infty(\Phi(q), \Phi(o)) > t$ we eliminate the object o from the search because the Lemma 2.4.2 guarantees that $\ell_\infty(\Phi(q), \Phi(o))$ is never greater than $d(q, o)$. In other words, $\ell_\infty(\Phi(q), \Phi(o))$ is a lower bounding of $d(q, o)$.

Definition 2.4.6 (Upper and Lower Bounds)**.** Let $f : (\mathcal{D}_1, d_1) \rightarrow (\mathcal{D}_2, d_2)$ a metric embedding. We say that d_2 is a *lower-bounding* of d_1 (or f is *contractive*) if

$$\forall x, y \in \mathcal{D}_1 \qquad d_1(x, y) \leq d_2(f(x), f(y)). \tag{2.56}$$

The distance d_2 is an *upper-bounding* of d_1 if

$$\forall x, y \in \mathcal{D}_1 \qquad d_1(x, y) \geq d_2(f(x), f(y)). \tag{2.57}$$

Definition 2.4.7 (Proximity preserving)**.** The mapping $f : (\mathcal{D}_1, d_1) \rightarrow (\mathcal{D}_2, d_2)$ is *proximity preserving* if

$$\forall x, y, z \in \mathcal{D}_1 \qquad d_1(x, y) \leq d_1(x, z) \Rightarrow d_2(f(x), f(y)) \leq d_2(f(x), f(z)). \tag{2.58}$$

If a the metric transformation is contractive then a range query can be directly performed in the projected space; similarly, if it is proximity preserving then a k-NN query can be directly performed in the projected space. However, as pointed in [66]: *"most current algorithms for NN-queries are based in range queries, and with some care they can be done in the projected space if the mapping is contractive, even if it is not proximity preserving"*.

A measure of how good is a metric space $(\mathcal{D}_2, d_2)$ in approximate another space $(\mathcal{D}_1, d_1)$ can be provided in term of *distortion*:

Definition 2.4.8 (Distortion)**.** The distortion for an approximation $(\mathcal{D}_2, d_2)$ of a space $(\mathcal{D}_1, d_1)$ mapped by a function $f : (\mathcal{D}_1, d_1) \rightarrow (\mathcal{D}_2, d_2)$ is the smallest D such that, for some scaling factor r

$$\forall x, y, z \in \mathcal{D}_1 \qquad r \cdot d_2(f(x), f(y)) \quad \leq \quad d_1(x, y) \quad \leq \quad D \cdot r \cdot d_2(f(x), f(y)) \tag{2.59}$$

A special case of proximity preserving mappings with distortion equal to one, is that corresponding to projection that preserve all the actual interpoint distances:

Definition 2.4.9 (Isometric embedding)**.** The mapping $f : (\mathcal{D}_1, d_1) \rightarrow (\mathcal{D}_2, d_2)$ is an *isometric embedding* if $d_1(x, y) = d_2(f(x), f(y))$, for all $x, y \in \mathcal{D}_1$.

Clearly when the metric embedding is isometric then searching in the projected space is equivalent to searching in the original space, whatever query paradigm is used.

2.4.8.1 Finite Isometric Embeddings

Now we focus on a particular class of metric embedding which preserve the distance between a finite set of data objects. We used this class of transformations in Chapter 5 to improve the search in the so-called supermetric spaces.

Definition 2.4.10 (Finite isometric embedding)**.** A *finite isometric embedding* of one metric space $(\mathcal{D}_1, d_1)$ in another $(\mathcal{D}_2, d_2)$ is achieved when for any finite selection $\mathcal{X}$ of points from $\mathcal{D}_1$ ($\mathcal{X} \subset \mathcal{D}_1, |\mathcal{X}| < \infty$) exists a mapping function $f : (D, d_1) \rightarrow (\mathcal{D}_2, d_2)$ such that $d_1(x, y) = d_2(f(x), f(y))$, for all $x, y \in \mathcal{X}$.

We say that $\mathcal{D}_1$ is isometrically *n-embeddable* in $\mathcal{D}_2$ if this property is true for any finite selection of n points from $\mathcal{D}_1$:

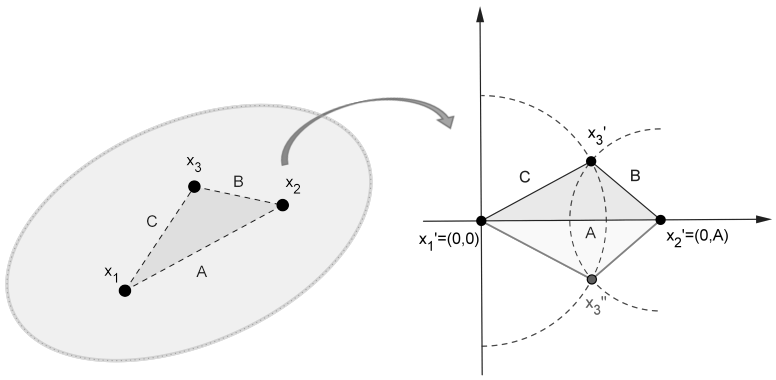


Figure 2.19: *For any three points x_1, x_2 and x_3 whose distances satisfy the triangle inequality property, a triangle can be constructed within 2D Euclidean space such that $\mathbf{x}'_1$ is at the origin, $\mathbf{x}'_2$ lies on the X-axis, and $\mathbf{x}'_3$ is where the distances B and C intersect. From [74].*

Definition 2.4.11 (Isometric n-embeddability)**.** A metric space $(\mathcal{D}_1, d_1)$ is *isometrically n-embeddable* into $(\mathcal{D}_2, d_2)$ if for any n points $o_1, \ldots, o_n \in \mathcal{D}_1$ there exists a function $f : (D, d_1) \to (\mathcal{D}_2, d_2)$ such that

$$\forall\, i, j = 1, \ldots n \qquad d_1(o_i, o_j) = d_2(f(o_i), f(o_j)). \tag{2.60}$$

The idea of characterizing a space metrically by means of "*n-point relations*" seems to have originated in the paper [86] published in 1892 by de Tilly, a Belgian artillery officer. Some of the question raised by de Tilly were answered by some mathematicians of the late 19th century, but only in 1928 Karl Menger [182] provided a first systematic development of abstract *distance geometry*. As highlighted by Blumenthal [51], "*distance geometry may operate in any kind of space in which a notion of "distance" is attached to any point-pair of the space*". The main interest of the distance geometry is in all those of transformations of sets for which the distance between two points is invariant.

The first observation to be made in this context is that any metric space is isometrically 3-embeddable in 2D Euclidean space (ℓ_2^2). This means that any three points of a metric space can be represented in two-dimensional Euclidean space while preserving all the three interpoint distances (see Figure 2.19). The 3-embeddability in 2D Euclidean space is apparent from the triangle inequality property of a proper metric. In fact, the two properties are equivalent: for any semimetric space which is isometrically 3-embeddable in ℓ_2^2, triangle inequality also holds.

It turns out that many useful metric spaces have a stronger property: they are isometrically 4-embeddable in 3D Euclidean space (ℓ_2^3), *i.e.* any four points of the space can be represented in a three-dimensional Euclidean space while preserving all the $\binom{4}{2} = 6$ interpoint distances. In the mathematical literature, this has been referred to as the *four-point property* [51]. Wilson [262] shows various properties of such spaces, and Blumenthal [51] points out that results given by Wilson, when combined with work by Menger [182], generalise to show that some spaces have the *n-point property*: that is any n points can be isometrically embedded in a Euclidean $(n-1)$-dimensional space).

The most important results in finite isometric embeddings from the perspective of our work are given by Schoenberg and Blumenthal. Schoenberg [229] shows that if a kernel function has certain simple properties, then it can be used to construct a metric space which is isometrically embeddable in a Hilbert space [9]. Blumenthal [51] shows that any space which is isometrically embeddable in a Hilbert space has the n-point property for every possible integer n. In combination these are extraordinarily strong from our perspective: for any kernel function with the correct properties, we can construct a proper metric space with the n-point property. We have studied such spaces in the context of metric search where we

[9] A Hilbert space H can be thought of as a generalization of Euclidean space to any finite or infinite number of dimensions. It is an inner vector space which is also a *complete* metric space with respect to the distance function induced by the inner product. This means that every Cauchy sequence in (H, d) converges to a point in H (intuitively, there are no "points missing" from H).

show how the n-point embeddings can be used to improve metric indexing and searching. We expand on this observation in Chapter 5.

2.4.9 Permutation-Based Approximate Similarity Search

The precise processing of a similarity query can be relatively expensive in terms of computational cost. In fact, if the search space is very large or it has a high intrinsic dimensionality, the exact methods rarely outperforms the sequential scan [190, 261]. Thus, an approximate search is likely to be of use in cases where the exact search is intractable. Algorithms for approximate search can be much more efficient than exact ones, at the expense of some accuracy reduction. However, controlled imprecision is totally tolerated in many applications, such as in multimedia retrieval where the notion of "similarity" may differ on the user' expectations and close approximations may be good enough for human perception [99].

Approaches to approximate search can be broadly divided into two main categories [99, 271]: (i) the ones based on reducing the subset of data that needs to be examined; (ii) the ones which exploit some space transformations.

The first category encompasses *early termination strategies* (*e.g.* stopping the search algorithm after a certain percentage of the dataset has been accessed, or after a specified time has elapsed) and *relaxed branching strategies* (*e.g.* use a "relaxed" pruning rule to avoid to access to region with a low likelihood to contains solutions to the query).

In the second category, the approximation is given by changing the object representation and/or the distance function in order to work in a more tractable space. A typical example is using a *dimensionality reduction* methods, such as PCA. Metric transformations that provide lower bounds on the actual distance are preferred when dealing with range queries. Metric transformations that are proximity preserving are preferred when dealing with nearest neighbour queries. In facts, in such cases even if the result set contains "false hits", it is still possible to obtain the exact result set by subsequently refining the approximate results according to the original distance function. Of course, the request is that the searching in the embedded space is much more efficient than searching in the original space (*e.g.* embedding a space with the Jensen-Shannon Distance into a small dimensional Euclidean space with the technique that we propose in Section 5.4). However, there exist many approximate approaches using metric transformations that are not distance-preserving or that do not satisfy the lower bounding property. In that cases, the desire is that the transformed space somehow preserve the proximity of the original objects and that it can be efficiently indexed and searched. Successfully example are the *LSH* model [136] within its extensions [46, 174] and *Permutation-Based Indexing (PBI)* [36, 64, 96] approaches. In the following, we introduce some measures of performance typically used to evaluate the efficiency and effectiveness of approximate algorithms. We then review principles of PBI.

2.4.9.1 Performance Measures for Approximate Search

The main goal of approximate similarity search is to reduce search time for answering a similarity query by introducing a possibly "small" error in the results. Thus, performance assessments

The *improvement in efficiency* of an approximate algorithm with respect to an exact search is expressed as the ratio between the cost of the exact and the approximate execution of a query. This is mainly applied to early termination strategies. The cost may denote the number of disk accessed, or the number of distance evaluation, to execute a query.

Several measures for assessing the accuracy of an approximation technique have been proposed in the literature (see [271, ch.9]). We focus on *precision* and *recall*, introduced in Section 2.1.1, which are widely used in information retrieval. In the context of approximate similarity search, the results-set of the exact search play the role of a ground-truth. So, if $\mathcal{R}$ denotes the results-set of the exact search and $\mathcal{R}^A$ the result-set returned by the approximation algorithm, we have

$$precision = \frac{|\mathcal{R} \cap \mathcal{R}^A|}{|\mathcal{R}^A|} \tag{2.61}$$

and

$$recall = \frac{|\mathcal{R} \cap \mathcal{R}^A|}{|\mathcal{R}|}. \tag{2.62}$$

Please note that in this context the interpretation of precision/recall may be misleading with respect their typical use in information retrieval. For example, for a range query, may often happen that $\mathcal{R}^A \subseteq \mathcal{R}$ and thus that precision is always 1 (*e.g.* when using contractive mappings). In this case, the precision gives no useful information. Moreover, in the case ok a k-NN query the precision equals the recall since the exact and the approximate result set have cardinality k. In such cases we will use the terminology *recall@k*:

$$recall@k = \frac{|\mathcal{R}(k) \cap \mathcal{R}^A(k)|}{k} \tag{2.63}$$

where $\mathcal{R}(k)$ and $\mathcal{R}^A(k)$ are the top-k sets of results returned by exact and approximate similarity search, respectively.

2.4.9.2 Permutation-Based Indexing

Permutation-based methods are characterized by representing each data object as a sequence of identifiers (*permutation*) in such a way that similar objects have similar permutations. Similarity queries are executed by searching for data objects whose permutation representation is similar to the query one.

Typically, the permutation is computed as a ranking list of some preselected reference points, called *pivots*, according to their distance to a given object. In this case, the similarity between objects is assessed on the basis of their relative distance to pivots.

The idea of approximating the distance between any two objects of a metric space by comparing their permutation-based representation was originally proposed in [38, 64]. The main rationale behind this proposal is that if two objects are very close one to the other, they will sort the set of pivots in a very similar way, and thus the corresponding permutation representations will be close as well.

Permutation-based searching belongs to a class of *filter-and-refine* methods: given a query, the search is firstly performed on a simplified representation of the original data (the permutation) in order to efficiently find a list of candidate results; the candidate set can be then refined by directly comparing the candidate objects with the query, using the distance in the original space. Search methods based on permutations mainly differ in the way of producing the candidate set (*i.e.* the filtering step). One straightforward way is the *brute-force searching* in the permutation space. However, this approach is not feasible on very large scale. Furthermore, when the distance in the original space is cheap to be computed (e.g., Euclidean distance), the brute-force search in the permutation space is not much faster than the brute-force search in the original space [190]. Several techniques for *indexing* and searching of permutations were proposed, including (i) index permutations using inverted files, like the *Metric Inverted File (MI-File)* [36] and its modifications (*e.g.* the *Neighborhood APProximation index* (NAPP) [241]); (ii) using prefix tree, like the *Permutation Prefix Index (PP-Index)* [96], the *Pivot Permutation Prefix Index (PPP-Index)* [200] and the *M-index* [198]; (iii) other existing index methods for generic metric space [100].

In [24], various pivot selection techniques (namely, random selection, k-medoids [151], Balancing Pivot-Position occurrences (BPP) [24], Farthest-First Traversal (FFT) [111], and Pivoted Space Incremental Selection (PSIS) [60]) were tested on three permutation-based indexing/searching approaches (*i.e.* MI-File, PP-Index, and linear scan of the permutations). The results revealed that each indexing approach has its own best selection strategies but also that the random selection of pivots, even if never the best, results in good performance.

In [109] a Surrogate Text Representation (STR) derived from the permutation representation has been proposed. The conversion of the permutations in a textual form allows using off-the-shelf text search engines to perform similarity search.

Permutation-Based Representation Let $\mathcal{D}$ a domain of data objects (*e.g.* image features), and let $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}^+$ a dissimilarity function, such as a metric distance. Given a fixed set of pivots $P = \{p_1, \ldots, p_n\} \subset \mathcal{D}$ we define a permutation-based representation Π_o (briefly *permutation*) of an object $o \in \mathcal{D}$ as the sequence of pivots identifiers ordered by their similarity to o.

Formally, the permutation $\Pi_o = [\Pi_o(1), \Pi_o(2), ..., \Pi_o(n)]$ lists the pivot identifiers $\{1, \ldots, n\}$ in an order such that $\forall\, i, j \in \{1, \ldots, n\}$

$$\Pi_o(i) < \Pi_o(j) \Leftrightarrow \big(d(o, p_{\Pi_o(i)}) < d(o, p_{\Pi_o(j)}) \big) \vee \big(d(o, p_{\Pi_o(i)}) = d(o, p_{\Pi_o(j)}) \wedge (i < j) \big) . \tag{2.64}$$

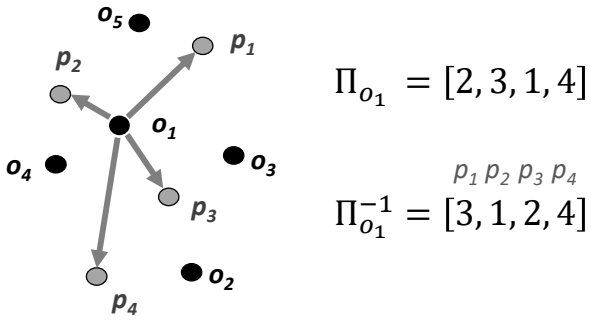


Figure 2.20: *Example of permutation-based representations of an object, given a set of four reference points. Black points are data objects. Grey points are reference objects.*

In other word, $p_{\Pi_o(j)}$ is the pivot at position j in the ranked list of the nearest pivots to the the object o. For example, given four pivots p_1, p_2, p_3, p_4 and an object o, if $\Pi_o = [2, 3, 1, 4]$ then p_2 is the nearest pivot to o, p_3 is the 2nd nearest pivot, and so on.

By denoting as $\Pi_o^{-1}(i)$ the position of a pivot p_i in the permutation Π_o, we obtain an equivalent representation $\Pi_o^{-1} = [\Pi_o^{-1}(1), \Pi_o^{-1}(2), \ldots, \Pi_o^{-1}(n)]$, called *inverted permutation*, which satisfies $\Pi_o(\Pi_o^{-1}(i)) = i$. The inverted permutation is very useful for essentially one reason: it allows us to represent permutations in a Cartesian coordinate system and easily define most of the commonly-used distances between permutations as distances between Euclidean points.

In some applications just the l-nearest pivots are used to represent an object, with $l < n$. In this cases, the *truncated permutation* (or *permutation prefix*), defined as $\Pi_{o,l} = [\Pi_o(1), \ldots, \Pi_o(l)]$, and the *inverted truncated permutation*, defined as

$$\Pi_{o,l}^{-1}(i) = \begin{cases} \Pi_o^{-1}(i) & \text{if} \quad \Pi_o^{-1}(i) \leq l \\ l+1 & \text{otherwise} \end{cases} \tag{2.65}$$

are used.

In summary, the value at the coordinate i in the permutation Π_o is the identifier of the pivot at i-th position in the ranked list of the nearest pivots to o. In the inverted representation Π_o^{-1}, each coordinate dimension corresponds to a pivot and so the value at the coordinate i is the rank of the pivot p_i in the list of the nearest pivots to o (see also Figure 2.20).

Please note that the computation of the distances between pivots and the data objects is just one, yet effective, approach to associate a permutation to each data object. For example, in Chapter 4 (Sec.4.2) we propose an effective and efficient approach to generate permutations for indexing Convolutional Neural Network features, without computing any object-pivot distances.

Distances between permutations Statisticians have used a number of different measures of closeness for ranked lists. The works of Kendall and Gibbons [152] and Diaconis [91] provide a formal treatment of this topic. The most used metrics to compare permutations in the context of PBI are the *Kendall's tau*, *Spearman's footrule*, *Spearman's rho* and the extensions of these metrics to partially ranked list (*i.e.*, *top-l distances*) [80, 97].

Definition 2.4.12 (Kendall's tau (K_τ))**.** The *Kendal's tau* distance between two permutation Π_x, Π_y on n elements equals the minimum number of pairwise adjacent transpositions to convert one permutation to the other. Formally,

$$K_\tau(\Pi_x, \Pi_y) = \sum_{i,j=1}^{n} K_{i,j}(\Pi_x, \Pi_y) \tag{2.66}$$

where

$$K_{i,j}(\Pi_x, \Pi_y) = \begin{cases} 0 & \text{if} \quad \left(\Pi_x^{-1}(i) - \Pi_x^{-1}(j)\right)\left(\Pi_y^{-1}(i) - \Pi_y^{-1}(j)\right) > 0 \\ 1 & \text{otherwise} \end{cases}. \tag{2.67}$$

Definition 2.4.13 (Spearman's footrule (S_F))**.** Given two permutations Π_x, Π_y on n elements, the *Spearman's footrule* is defined as

$$S_F(\Pi_x, \Pi_y) = \sum_{1 \leq i \leq n} (\Pi_x^{-1}(i) - \Pi_y^{-1}(i)) \tag{2.68}$$

Definition 2.4.14 (Spearman's rho (S_ρ))**.** Given two permutations Π_x, Π_y on n elements, the *Spearman's rho* is defined as

$$S_\rho(\Pi_x, \Pi_y) = \sqrt{\sum_{1 \leq i \leq n} \left(\Pi_x^{-1}(i) - \Pi_y^{-1}(i)\right)^2} \tag{2.69}$$

The majority of PBI methods, e.g. [36, 96, 189, 198, 200], do not use the full-length permutations in practice, but rather use only the nearest reference points to represent and compare data objects (*i.e.* using the permutation prefixes typically in conjunction with *top-l distances* [97]). These approaches, on one hand, follow the intuition that the most relevant information of a permutation is in the very first, *i.e.* nearest, pivots. On the other hand, they are based on experimental results showing that using positions of the *top* ℓ out of n pivots often lead to obtain better or similar effectiveness than using the full-length permutation, resulting also in a more compact data encoding.

Example of top-l distances are the *Spearman's rho* and the *Spearman's footrule with location parameter l*, which were initially proposed in [97] to compare partial ranked list (a list that contains rankings for only a subset of items).

Definition 2.4.15 (Spearman's footrule/rho with location parameter l ($S_{F,l}$ / $S_{\rho,l}$))**.** Given two permutations Π_x, Π_y on n elements, and theirs truncated permutations for an integer $l < n$, we define the *Spearman's footrule with location parameter l* as

$$S_{F,l}(\Pi_x, \Pi_y) = \sum_{1 \leq i \leq n} (\Pi_{x,l}^{-1}(i) - \Pi_{y,l}^{-1}(i)). \tag{2.70}$$

Similarly, the *Spearman's rho with location parameter l* is defined as

$$S_{\rho,l}, (\Pi_x, \Pi_y) = \sqrt{\sum_{1 \leq i \leq n} \left(\Pi_{x,l}^{-1}(i) - \Pi_{y,l}^{-1}(i)\right)^2}. \tag{2.71}$$

Note that only the first l elements of the permutations are needed, in order to compare any two objects with the $S_{\rho,l}$ or $S_{F,l}$.

Permuthahedron Since the inverted permutation representation Π_x^{-1} belongs to $\mathbb{R}^n$, the Spearman's rho between two (truncated) permutations corresponds to the Euclidean distance between the corresponding inverted (truncated) permutations. Similarly, the Spearman's footrule distance corresponds to the Manhattan distance between the inverted (truncated) permutations. Thus, the use of the inverted representations allows us to represent the permutations of the pivots identifiers in a Cartesian coordinate system.

For example, let consider all possible permutation-based representation of $n = 3$ objects (*i.e.*, the set of all permutations on 3 elements): $\{[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]\}$. It is easy to see that all these points lie on the plane $x + y + z = 6$ and represent the vertices of a regular hexagon as depicted in Figure 2.21.

Consider now the $n = 4$ case: the vectors of all possible Π_o^{-1} lie in a three-dimensional subspace of $\mathbb{R}^4$ and are the vertices of a truncated octahedron (see Figure 2.22).

In general, the $n!$ points x obtained by permuting the coordinates of the vector $[1, 2, \ldots, n]$, form the vertices of a $(n-1)$-dimensional polytope embedded in a n-dimensional space, referred to as *permutahedron* (also spelled *permutohedron*) [107, 280]. In particular, given that both the sum of vector values x_i (*i.e.*, $\Pi_o^{-1}(i)$) and the sum of their squared values are fixed, all the vertices of the permutahedron lie on both the hyperplane of equation

$$x_1 + x_2 + \cdots + x_n = \frac{n(n+1)}{2}$$

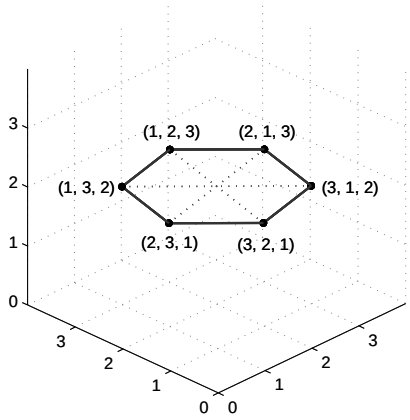


Figure 2.21: *The six points in $\mathbb{R}^3$ obtained by permuting the coordinate of the vector $(1, 2, 3)$. From [31].*

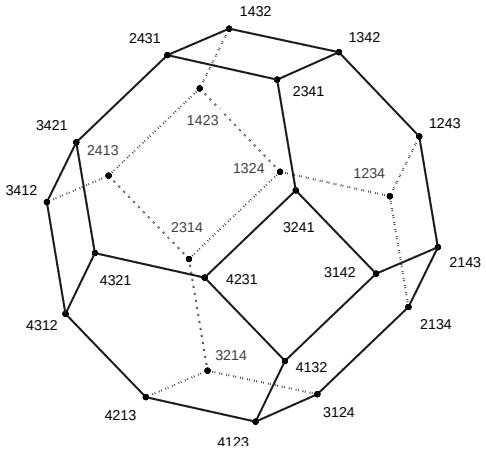


Figure 2.22: *Permutahedron with $4! = 24$ vertices. From [31].*

the $n-$sphere of equation

$$x_1^2 + x_2^2 + \cdots + x_n^2 = \frac{n(n+1)(2n+1)}{6}.$$

That is they lie on a $n-1$ sphere residing in n-dimensional space given by the intersection between an hyperplane and a sphere both in $\mathbb{R}^n$ [228].

The permutahedron is a very interesting convex polytope. It is centrally symmetric and its vertices can be identified with the permutation of n objects in such a way that two vertices are connected by an edge if and only if the corresponding permutations differ by an adjacent transposition. It is rather easy to see that the squared Euclidean distance between any two vertices is an even integer, moreover, for $n > 4$, the squared distances constitute every even integer up through the maximum possible value, that is $\frac{1}{3}(n^3 - n)$ [228, 280].

As observed in [228], standing on any vertex of a permutahedron and looking around at neighbouring vertices, the view of the surrounding space is the same: there would be $n-1$ adjacent vertices evenly distributed around the observation vertex, which Euclidean distance is $\sqrt{2}$. Furthermore, the number of vertices and their relative positions within a generic r-ball neighbourhood is independent of the observation vertex. This is not true in practice: Skala [234] proved that not all the permutations actually exist in the permutation-based encoding defined by a given preselected pivots.

Other observations on PBI Choosing the total number n of the pivots and the prefix length l of the truncated encodings is an open issue. They seem to depend on the size of the dataset, its intrinsic dimensionality and on the distance used to compare the permutations. Some heuristics were proposed in the literature. Amato et al. [36] proposed to use $n > 2\sqrt{N}$ reference objects, where N is the size of the dataset. They used the Spearman's footrule distance with location parameter l ($S_{F,l}$) and experimentally observed that if the dataset is clustered, the optimal parameter l is around N/cl, where cl is the number of clusters in the dataset.

In [189], Mohamed and Marchand-Maillet also used $S_{F,l}$ and empirically derived that with their Metric Suffix Array (MSA) the best ratio between the number of nearest pivots and the dataset is $l = \sqrt{N/2}$ and the closest pivots can be chosen out of $n = 2l$ reference points.

In general, choosing the total number n of pivots and the optimal length l highly depends also on the used index. For example, the MI-File [36] benefits from using high value for n and relatively small number l (*e.g.* $n = 2,000$ and $\ell = 100$ to index 1M-CoPhIR dataset [54]), while M-index [198], PP-Index [96] and PPP-Index [200] achieve good recall using short prefix and relatively few pivots (*e.g.* $n = 100/200$ and $l = 6/8$ to index 1M-CoPhIR dataset) but they typically build several copies of the index using different set of pivots (*e.g.* 4 indexes).

2.4.9.3 Surrogate Text Representation

In [109], Gennaro et al. firstly proposed a technique, called Surrogate Text Representation (STR), that exploits the permutation-based representation to associate an object with a text encoding. As pointed in [19], "*the objective is to define a function that transforms a global descriptor into a sequence of terms (*i.e. *a text document) that can be fed into a text search engine as for instance Lucene. Of course, the ultimate goal is to obtain that the distance between the documents and the query is an approximation of the original distance function of the global descriptors*". Thus the main advantage of the text encoding of a metric object is that any off-the-shelf text retrieval engine can be used to perform similarity search.

The textual representation is obtained by associating each pivot with a unique alphanumeric keyword and each object with a sequence of the keywords, built on the basis of the permutation-based representation of the object. Specifically, given a set of n pivots, each object o is first encoded as a permutation Π_o of the pivot identifiers. Each pivots p_i is then associated to a unique keywords τ_i (*e.g.*, τ_1 ="A", τ_2 ="B", etc.) and each object permutation Π_o is transformed into a text document t_o composed as the concatenation of some keywords. The text is build in such a way that the occurrence of the keyword τ_i in t_o reflects the closeness of the pivot p_i to the object o. So, if considering the inverted permutation Π_o^{-1} we have that the lower the value $\Pi_o^{-1}(i)$ the higher the frequency of the term τ_i in the document t_o. Formally, the text document is a space-separated concatenation of one or more keywords as follows:

$$\mathrm{t}_o = \bigcup_{i=1}^{n} \bigcup_{j=1}^{n-\Pi_o^{-1}(i)} \tau_i, \tag{2.72}$$

where, by abuse of notation, we denote the space-separated concatenation of keywords with the union operator $\cup$. For example, given an inverted permutation $\Pi_o^{-1} = [3, 1, 4, 2]$ and the codebook $\{$ τ_1 ="A", τ_2 ="B", τ_3 ="C", τ_4 ="D"$\}$ we obtain that in t_o the frequency of "A" is $4 - 3 = 1$, the frequency of "B" is $4 - 1 = 3$, and so on, obtaining t_o ="B B B D D A".

The next step is index the transformed objects with inverted files. In order to reduce the size of the inverted index, just $l < n$ closest pivots are used for indexing, *i.e.* a location parameter is considered. In this case the text encoding is

$$\mathrm{t}_{o,l} = \bigcup_{i=1}^{n} \bigcup_{j=1}^{l+1-\Pi_{o,l}^{-1}(i)} \tau_i. \tag{2.73}$$

Typically, two different location parameters are used: l_o for indexing and l_q for querying since the performance of the inverted files is optimal when the size of the queries are much smaller than the size of the documents. Thus, the general requirement is $l_q < l_o$.

The resulting text representations are then indexed and searched by using conventional text search engine, such as Lucene, that typically use Cosine Similarity to measure the matching degree of a query vector with document vectors. The justification is that, given two objects o and q, the Cosine Similarity between the textual representation t_o and t_q reflects the similarity between the permutations Π_o and Π_q if the Spearman rho distance is used for comparing the permutations. We provide a formal proof of this in Section 4.1.1, where we also propose an extension of the baseline STR for effectively encoding VLAD features. In summary, the baseline STR representation is equivalent to the permutation representation with the Spearman rho distance. The main advantage of STR with respect to the permutations is that it gives us the possibility of exploit high-performance text search engine library with little implementation effort.

2.5 Datasets

In this Section, we summarize datasets used in the later chapters (grouped on the basis of their use in *this* thesis).

2.5.1 Datasets used for Retrieval and Recognition Tasks

Here we report image collections for which a ground-truth is provided, or it has been manually built from us. The ground-truth is used to measure the retrieval performance of various tested approaches.

INRIA Holidays [139] is a collection of $1,491$ images which mainly contains personal holidays photos. The images are of high resolution and represent a large variety of scene type (natural, man-made, water, fire effects, etc). The dataset contains 500 queries, each of which represents a distinct scene or object. For each query, a list of positive results is provided. Some example images are shown in Figure 2.23. This dataset is used in the experiments of Chapters 3 and 4.

Figure 2.23: *Example photos from the INRIA Holidays dataset.*

Oxford5k [211] consists of $5,062$ images collected from Flickr. The dataset comprise 11 distinct Oxford buildings together with *distractors*. There are 55 query images: 5 queries for each building. The collection is provided with a comprehensive groundtruth. For each query there are four image sets: *Good* (clear pictures of the object represented in the query), *OK* (images where more than 25% of the object is clearly visible), *Bad* (images where the object is not present) and *Junk* (images where less than 25% of the object is visible or images with high level of distortion). Some examples of the collection are shown in Figure 2.24. This dataset is used in the experiments of Chapter 3.

Figure 2.24: *Example photos from the Oxford5k collection.*

Epigraphic Database Roma (EDR) [3] is part of the international federation of Epigraphic Databases called Electronic Archive of Greek and Latin Epigraphy (EAGLE). EAGLE federation's purpose is to collect all published Greek and Latin inscriptions up to the 7th century A.D., while EDR aims at collecting the whole epigraphy of Rome and of the Italian peninsula. Presently EDR has records for $82,921$ inscriptions and contains $54,374$ photos. In Chapter 3 we describe a subset of $17,155$ photos related to $14,560$ inscriptions that we had access and for which we have manually built a ground-truth. Some example images are shown in Figure 2.25.

Figure 2.25: *Example images from EDR collection.*

Pisa Dataset [10] is composed of $1,227$ photos of 12 monuments and landmarks located in Pisa. This dataset was created during the VISITO Tuscany project[10] and the photo were crawled by Flickr [11]. It is divided into a training set consisting of 226 photos (20% of the dataset) and a test set consisting of 921 photos (80% of the dataset). Some example photos are shown in Figure 2.26. This dataset is used in the experiments of Chapters 3.

Figure 2.26: *Example photos from the Pisa dataset, uploaded to Flickr by the following users (left to right): LivornoQueen, eddip51, allylic, and Bunburyshire.*

2.5.2 Datasets Used for Training

The following data collections are used in our experiments for various learning stages (*e.g.*, learning a visual vocabulary).

Flickr60k [139] is composed of $67,714$ images extracted *randomly* from Flickr. This dataset is used in the experiments of Chapters 3 as training set for INRIA Holidays dataset. Some examples of the collection are shown in Figure 2.27.

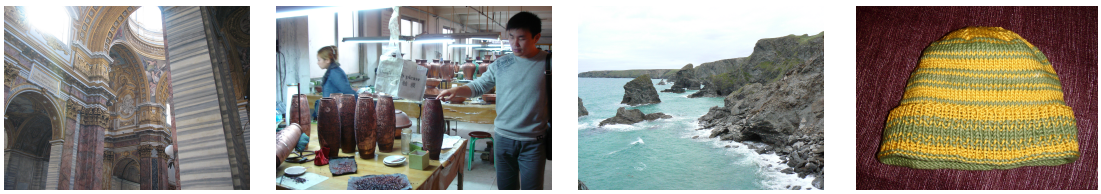


Figure 2.27: *Example photos from the Flickr60k dataset, uploaded to Flickr by the following users (left to right): Don Stein, royare, Rob Phillips, and thegloaming.*

Paris6k [212] contains $6,300$ high resolution images collected from Flickr by searching for famous Paris landmarks. We used this dataset in the experiments of Chapters 3 as a training set for the Oxford5k dataset, as proposed by the authors of the dataset. Some example images are shown in Figure 2.28.

Figure 2.28: *Example photos from the Paris6k dataset. We downloaded the images from [9] where no information about the authors of the photos is provided.*

[10] http://www.visitotuscany.it/index.php/en

[11] Some photos are no longer available on Flickr, however we have access to whole dataset thanks to the collaboration with ISTI-CNR

2.5.3 Datasets Used for Similarity Search

Finally, we report datasets used in the context of similarity search.

Yahoo Flickr Creative Commons 100 Million (YFCC100M) dataset [243] contains almost 100M images, all uploaded to Flickr between 2004 and 2014 and published under a CC commercial or non-commercial license. We used this dataset in Chapter 4. It is worth noting that in our experiments we do not directly access to the images of this dataset, but we rather use the CNN features extracted by Amato et al. [28]. In fact, for these features there exists a ground-truth for exact similarity related to 1,000 queries [28]. Some example images are shown in Figure 2.29.

Figure 2.29: *Example photos from the YFCC100M dataset, uploaded to Flickr by the following users (left to right): Johan Fabry, Ed Mitchell, Denise Fasanello, Patrick Ashley, and Barney Loehnis.*

SISAP Colors and Nasa are two datasets that are commonly used as benchmarks for metric indexing approaches. The *Nasa* [8, 101] dataset contains 40,150 feature vectors, each of 20 real coordinates, generated from images downloaded from NASA photo and video archive site. The *Colors* [101] set contains 112,682 feature vectors of dimension 112, representing color histograms of medical images. We have not the access to the original images.

2.5.4 Other Datasets

Finally, we report other datasets used (or, implicitly used) in this thesis.

MIRFlickr [135] is a dataset of 1 million Flickr images. In literature, the MIRFlickr dataset is often used as distractor set for the INRIA Holidays data, and we do the same in Chapters 3 and 4. In [79], it was shown that MIRFlickr contains about $2,407$ pairs of near duplicate images, which makes this dataset also attractive for testing near-duplicate detection algorithms. With this respect we used MIRFlickr data in Chapter 5. Some example images are shown in Figure 2.30.

Figure 2.30: *Example photos from ILSVRC2012, uploaded to Flickr by the following users (left to right): Hugo Adolfo Beltran Olivas, Dave Wild, Martin P. Szymczak, and Silke Gerstenkorn.*

ILSVRC2012 is a subset of the famous ImageNet [88], which is a very large visual database organized according to the WordNet[12] hierarchy. Actually, it contains more than 15M images related to about 22K categories. The ILSVRC2012 is a subset of ImageNet created during the Large Scale Visual Recognition Challenge 2012. It contains about 1.5 million images and $1,000$ classes. Even if we do not use this

[12] `http://wordnet.princeton.edu/`

dataset explicitly, here we report it since it has been used as learning data for several pretrained CNN models that we use in Chapters 3 and 4.

Places205 is a subset of Places Database [276], which is very large scene-centric database. Places Database contains contain more than 7 million images from 476 place categories. Its subset Places205 contains 2.5 million images from 205 scene categories. As for ILSVRC2012, we do not explicitly use Places205, but in the experiments of in Chapters 3 and 4 we use some CNN models trained on it.

CHAPTER 3

Efficient and Effective Image Features

Performance of CBIR and recognition systems, in term of efficiency and effectiveness, highly depends on the features employed to represent the image visual content. Therefore, the research for "good" image descriptors has been the object of much interest from the research community. State-of-the-art image representations rely on *local features* (*e.g.* SIFT), *aggregations of local features* (*e.g.* FV), and *deep learning* (*e.g.* CNN features), introduced in Sections 2.1 and 2.3. Local features are at the core of many computer vision applications thanks to their robustness to geometric transformations and their effectiveness for matching local structures between images. However, representing and comparing images on the basis of local features is not efficient due to the complexity of extracting, storing, and matching the local descriptors. On one hand, aggregation techniques provide a meaningful summary of all the extracted features of an image into a single descriptor, allowing to speed up and scale up the image search. On the other hand, the cost for extracting, representing, and comparing local visual descriptors has been dramatically reduced by recently proposed *binary local features* (*e.g.* ORB). In order to achieve higher efficiency, few works have recently mixed together these two research directions by defining aggregation methods for binary local descriptors [108, 114, 163, 251, 257, 274]. In recent years, the popularity of the local feature-based methods has been overtaken by deep learning approaches, which have demonstrated impressive performance in many vision tasks. For example, Razavian et al. [217] showed that the activations produced by an image within the top layers of a CNN can be used as high-level descriptors, achieving state-of-the-art quality results for object image classification, scene recognition, fine-grained recognition, attribute detection and image retrieval. Compared to descriptors built upon local features, the CNN features seems to carry richer high-level semantic information. However, they are less robust to some geometrical transformations of the images[1], which are crucial to instance retrieval tasks. In [63] a fusion of FV and CNN features was investigated to improve retrieval results and balance the lack of geometrical invariance of the CNN features. Recently, other researchers exploited hybrid architecture in order to achieve higher effectiveness [207, 231, 239].

In this chapter, we investigate both issues related to effectiveness and efficiency of image features. First we focus on effective image features. Particularly, in Section 3.1 we compare effectiveness of FV, CNN features and their combination. Our evaluation is performed in an applicative scenario that is rec-

[1]The robustness of the CNN features to image transformations depends on the used architecture and the training of the model. In [63] it is shown that features extracted using some common *pre-trained models* are less robust to transformation such as rotation or scale changes than FVs build upon SIFTs.

ognizing ancient inscriptions and other objects of cultural heritage. Interestingly, combining FV and CNN features into a single image representation allows us to achieve very high effectiveness. In Section 3.2, we focus on efficiency by performing an extensive comparison among the state-of-the-art aggregation methods applied to binary features. Furthermore, we mathematically formalize the application of Fisher Kernels to binary features by proposing the BMM-FV, which is a Fisher Vector computed using a Bernoulli Mixture Model. Finally, we investigate the combination of the aggregated binary features with the CNN features. Interestingly, the combination of the CNN with our BMM-FV allowed us to obtain a relative improvement over the CNN results that are in line with that recently obtained in [63] using the combination of the CNN with the FV built upon SIFTs. The advantage of using the BMM-FV is that it relies on binary local features whose extraction process is about two orders of magnitude faster than SIFTs (*e.g.* for extracting 2000 features from an image ORB takes about 26ms, while SIFT needs more than one second [126]).

The research presented in this chapter was published in [18, 21, 34, 37, 53], as regards the evaluation of effective image descriptors and its application to recognize ancient inscriptions, and [32, 33, 35] for the research on the aggregations of binary local features and the definition of the BMM-FV.

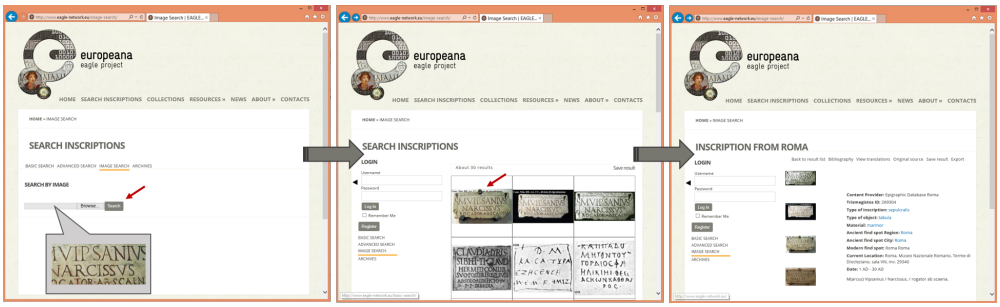
Our research activity was conducted in the context of the *Europeana network of Ancient Greek and Latin Epigraphy* (EAGLE) that is a best-practice network partially funded by the European Commission. One of the main motivations of the project was to collect in a single repository information about the thousands of Greek and Latin inscriptions presently scattered in a number of different institutions (museums and universities) across all Europe. The collected information, about 1.5 million digital objects (texts and images), represent approximately 80% of the total amount of classified inscriptions in the Mediterranean area. That information is ingested into Europeana and is also made available to the scholarly community and to the general public, for research and cultural dissemination, through a user-friendly portal. The EAGLE portal[2] supports advanced query and search capabilities, including visual search [12,18,37]. In order to increase the usefulness and visibility of its content, EAGLE has developed also a mobile application [53] that enables users (like tourists and scholars) to obtain detailed information about the inscriptions they are looking at by simply taking pictures with their smartphones and sending them to the EAGLE portal for recognition (*e.g.*, Figure 3.1). This represents a profitable and user-friendly alternative to the traditional way of retrieving information from an epigraphic database, which is mainly based on submitting text queries, for example, related to the place where an item was found or where it is currently located.

3.1 Evaluation of the State-of-the-Art for Visual Recognition

In this section, by discussing an applicative scenario, we compare several image representations that can be used to perform image retrieval and recognition. In particular, we thoroughly tested FV, CNN, and their combinations to visually recognize ancient inscriptions, such as Greek and Latin epigraphs. We then compare the obtained results with that we previously achieved in this context by using BoW and VLAD representations [30]. Our experiments, conducted on $17,155$ photos related to $14,560$ inscriptions, show that BoW and VLAD are outperformed by both Fisher Vector and CNN features. More interestingly, combining FV and CNN features into a single image representation allows us to achieve very high effectiveness by correctly recognizing the query inscription in more than 90% of the cases (rather than about 70% previously obtained with BoW and VLAD [30]). Our results suggest that combinations of FV and CNN can be effectively exploited to visually search other databases of objects, for example, related to cultural heritage (monuments, landmarks, paintings, etc.). In this respect, we also report results of FV and CNN features to search the Pisa Dataset [26] which contains photos of monuments and landmarks located in Pisa. The combination of FV and CNN leads to improve the retrieval performance also in this case.

[2] `https://www.eagle-network.eu`

(a) *Example of image search using the EAGLE Flagship Mobile Application (`https://www.eagle-network.eu/resources/flagship-mobile-app/`)*



(b) *Example of image search using using the EAGLE Web Portal (`https://www.eagle-network.eu/image-search/`)*

Figure 3.1: *An application that enables a user to get information about a visible inscription by taking a photo,* e.g. *by using the EAGLE Flagship Mobile Application (a), or by uploading a query image on the EAGLE Web Portal (b). The EAGLE visual search engine, developed by ISTI-CNR, retrieves the photographed object from a database that currently contains more than 1 million inscriptions and provides the related information to the user [18, 37, 53]. In the depicted example, the provided information includes the transcription of the inscription (*M(arcus) Vipsanius / Narcissus, / rogator ab scaena.*), the type of the inscription (*sepulcralis*), the type of the object (*tabula*), and its present location (Roma), just to cite some.*

3.1.1 Combining FVs and CNN Features

Recently, researchers have shown that CNN features and local features have complementary behaviour under some image transformations. For example, in [63] extensive experiments on benchmark dataset for image retrieval showed that FVs (computed from SIFT or SIFTPCA) are particularly robust to image rotation, while the CNN features (extracted by common pre-trained models) have limited level of rotation invariance. Furthermore, according to their experiments, the CNN features are less affected by small-scale changes than FVs. So, in order to leverage on the positive aspects of both these methods, Chandrasekhar et al. [63] proposed a fusion of FV and CNN features and other works [207, 231, 239] have started exploring

Inspired by the recent literature, in this thesis we evaluate the combination of FV and CNN features using the following approach. Each image is represented by a couple $(\boldsymbol{c}, \boldsymbol{f})$, where $\boldsymbol{c}$ and $\boldsymbol{f}$ are respectively the CNN descriptor and the FV descriptor of a given image. Then, we evaluate the distance d between two couples $(\boldsymbol{c}_1, \boldsymbol{f}_1)$ and $(\boldsymbol{c}_2, \boldsymbol{f}_2)$ as the convex combination of the ℓ_2 distances of the CNN descriptors (*i.e.* $\|\boldsymbol{c}_1 - \boldsymbol{c}_2\|_2$) and the FV descriptors (*i.e.* $\|\boldsymbol{f}_1 - \boldsymbol{f}_2\|_2$). In other words, we define

$$d\left((\boldsymbol{c}_1, \boldsymbol{f}_1), (\boldsymbol{c}_2, \boldsymbol{f}_2)\right) = alpha\, \|\boldsymbol{c}_1 - \boldsymbol{c}_2\|_2 + (1 - \alpha)\, \|\boldsymbol{f}_1 - \boldsymbol{f}_2\|_2 \tag{3.1}$$

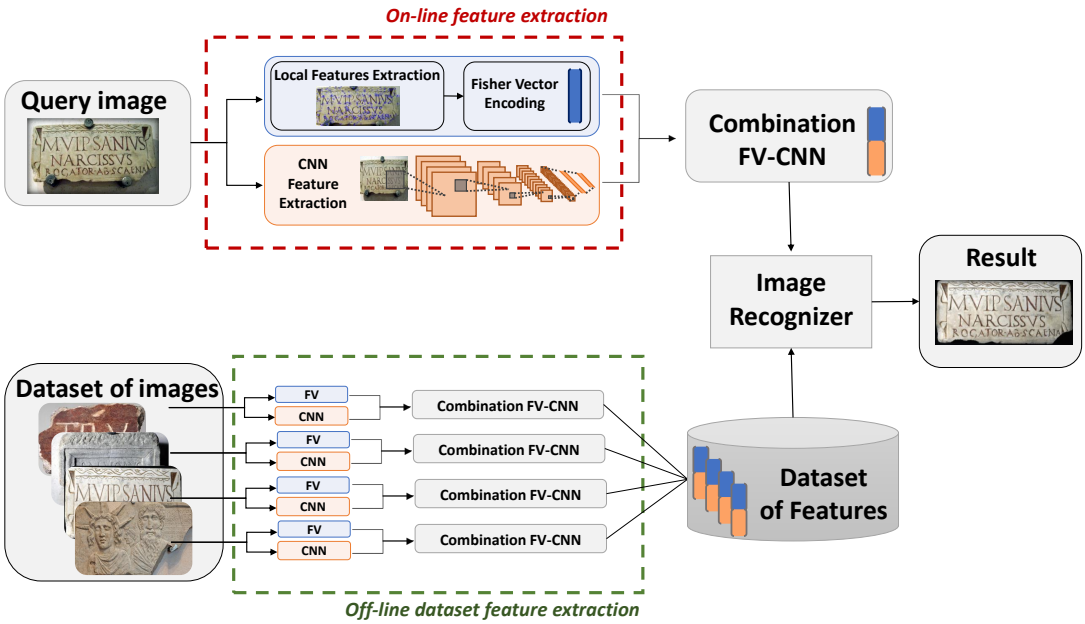


Figure 3.2: *Image recognition pipeline using the combination of FV and CNN*

with $0 \leq \alpha \leq 1$. Choosing $\alpha = 0$ corresponds to use only the FV approach, while $\alpha = 1$ corresponds to use only CNN features. In our case, both the FV and the CNN features are ℓ_2 normalized so the distance function between the CNN descriptors has the same range value of the distance function between the FV descriptors. Please note that combinations between CNN features and other image descriptors can be computed in a similar way by considering the convex combination of the respective distances. However, when the range of the two used distances is not the same, the distances should be rescaled before the convex combination (*e.g.* divide each distance function by its maximum value) in order to balance the contribution of each singular approach in the final result.

An example of visual recognition pipeline using FV and CNN is depicted in Figure 3.2.

3.1.2 Experiments on the Epigraphic Database Roma

Here we experimentally evaluate the state-of-the-art FV and CNN features, as well as their combinations, for recognizing ancient inscriptions. We first introduce the used dataset (Section 3.1.2.1) and other experimental settings (Sections 3.1.2.2 and 3.1.2.3). We then report results and their analysis (Sections 3.1.2.4).

3.1.2.1 Dataset and Ground Truth

Since ISTI-CNR was a partner of the EAGLE project, we had the opportunity to access and use a subset of the *Epigraphic Database Rome* (EDR) [3] (described in Section 2.5.1), which was made available to us by Sapienza University of Rome. The data subset we had access is composed of $17,155$ photos related to $14,560$ inscriptions, so in most cases just one photo is provided for each inscription object.

To carry out our performance analysis, we selected 70 queries, *i.e.* images to be recognized, and we manually built a ground truth for them. For each query, the ground truth contains all the images of the EDR dataset that were associated with the same object of the query, according to their metadata. During the retrieval tests we removed the query photo from the knowledge base. For this reason, we selected as query only inscriptions that have more than one photo in the dataset. Furthermore, the queries were carefully selected in order to represent the various types of inscriptions contained in the dataset (as, for example, inscriptions with a different state of preservation, or incised on different material). Figure 3.3 shows five query examples together with the corresponding relevant images. It is worth noting that the

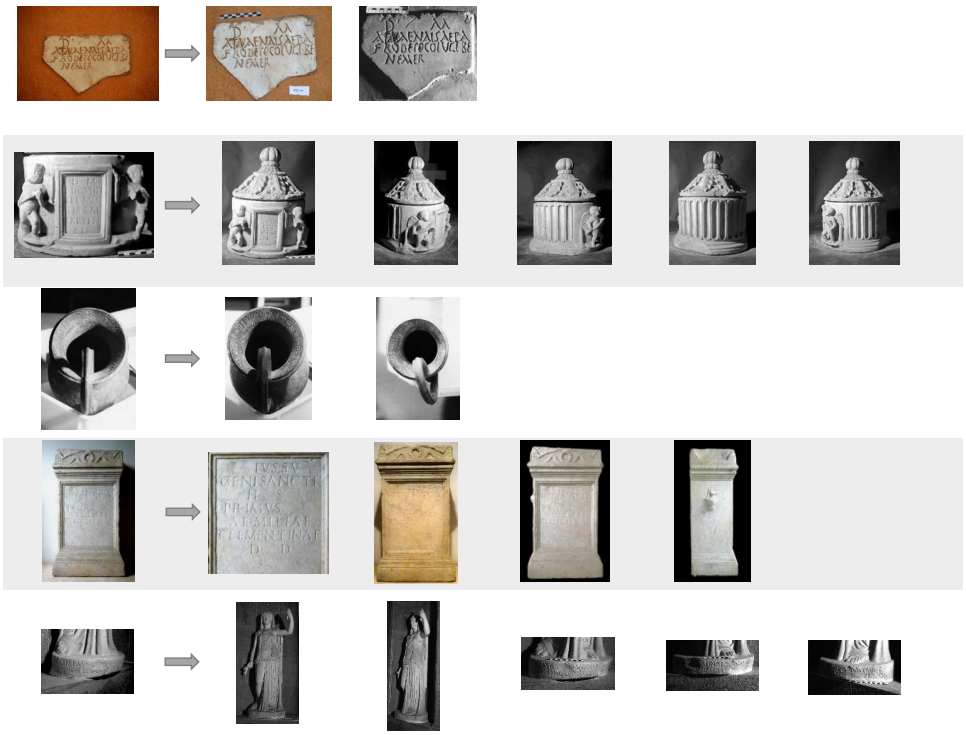


Figure 3.3: *Example of queries (on the right) and their associated images in the ground truth (on the left).*

EDR dataset (and so the ground truth) often contains very different viewpoints of the same object, which makes the retrieval of *all* the images relevant to a given query more challenging.

3.1.2.2 Performance Measures

In order to recognize the actual object in a query image, we perform a visual similarity search between all the images in the dataset. When examining the ranked result list of a query it is evident that the greater the ranked position of a relevant image the less valuable it is for the user, because the less likely the user will examine that image. The main goal is to have one relevant image as the first result. Whenever this is not the case, it is interesting to understand at which position in the result list a relevant image appears. In fact, in order to achieve better effectiveness, re-ranking techniques can be applied to the results list. Due to efficiency issues, the re-ranking is typically performed on a subset of the result list (the top-ranked results), which is the reason why we are interested in measuring the position of the first relevant image in the result list. Therefore we report, for each technique, the probability p of finding an image of the same query object within the first r results, varying r between 1 and 100.

The retrieval performance of each method was also evaluated by the mean Average Precision (mAP), with the query removed from the ranking list. During the mAP computation, not just the first relevant image but all the images associated with the query are considered. Therefore, while $p(r)$ measures how good each method is in reporting at least one relevant image in the first r positions of the result list, the mAP reveals how good each method is in reporting all the relevant images in the top positions.

Details on how $p(r)$ and the mAP are evaluated were described in Section 2.1.1.

3.1.2.3 Features Extraction

All the experiments were conducted using the publicly available Visual Information Retrieval library (VIR) [98] and other open source libraries. Here we report the details on the used image features and

how they were extracted.

Local features: We extracted SIFT local features by using OpenCV (Open Source Computer Vision Library) [56]. We obtained an average of 1,591 SIFT per image. However, the information about the scale at which the features were extracted allows us to select a subset of local features that are in principle more relevant. In fact, features detected at higher scale refer to bigger regions than others and should also be present at lower resolution versions of the same image or of the same object. Thus, the criterion that the bigger the scale the higher the importance can be used to perform feature selection, as proposed in [25]. In the experiments, we reduced the number of local features extracted from an image by selecting the 250 features detected at highest scales. We refer to the latter approach as *reduced-keypoints*. Clearly, the feature selection was not applied whenever the number of features extracted from an image was already less than 250.

Subsequently, the SIFT descriptors are reduced from 128 to 64 components by using PCA. The PCA rotation matrix was learned on about 2M of local features randomly selected from the whole dataset. We will see that the advantage of using PCA-reduced descriptors is twofold: the PCA decorrelates the dimension of the vectors, which is a wished property before the FV encoding, and decreases the memory footprint of the descriptors.

GMM and Fisher Vector: We implemented and integrated methods to compute the Gaussian Mixture Model and the Fisher Vector into the VIR [98], which is publicly available on GitHub. As commonly done in literature, and also described in Section 2.2.2.3, we estimate a GMM with the assumption of diagonal covariance matrices. So we consider a GMM of parameters

$$\lambda = \{w_k, \boldsymbol{\mu}_k = [\mu_{k1}, \ldots, \mu_{kD}], \boldsymbol{\Sigma}_k = \text{diag}(\sigma_{k1}, \ldots, \sigma_{kD})\}_{k=1,\ldots,K}$$

where K is the number of Gaussian, D is the dimension of each local descriptor, and w_k, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ are the mixture weight, the mean vector and the covariance matrix of the k-th Gaussian, respectively. The parameters in λ were learned by optimizing a maximum-likelihood criterion with the Expectation Maximization (EM) algorithm [50, ch.9]. EM is an iterative method that is deemed to have converged when the change in the likelihood function, or alternatively in the parameters to be estimated, falls below some threshold ϵ. As stopping criterion for the GMM estimation we used the convergence in ℓ_2-norm of the mean parameters, choosing $\epsilon = 0.05$. As suggested in [50, ch.9], the GMM parameters used in EM algorithm were initialized with: (a) $1/K$ for the mixing coefficients w_k; (b) centroids precomputed using k-means for the GMM means $\boldsymbol{\mu}_k$; (c) mean variance of the clusters found using k-means, for the diagonal elements σ_{kd} of the GMM covariance matrices.

Both the k-means and the GMM estimation were performed using in order of 1M local descriptors randomly selected from the whole dataset. As a common post-processing step [144, 209], the FVs were power-law normalized and subsequently ℓ_2-normalized. We recall that he power-law normalization is parametrized by a constant β and it is defined as $\boldsymbol{x} \rightarrow |\boldsymbol{x}|^{\beta}\text{sign}(\boldsymbol{x})$. In our experiments we used the common choice of $\beta = 0.5$.

We evaluated the performance of FV approach for various settings, summarized as follows.

- *SIFT or SIFTPCA64:* We built the FV both using full-size SIFT local descriptor (D=128) and SIFT reduced to 64 dimensions by means of PCA (SIFTPCA64).
- *Reduced-keypoint or not:* We tested the scale selection of the local features, as described above.
- *FV or FV$_\mu$:* We evaluated the performance of the whole FV as well as the FV related just to the mean vectors (as defined in Equation 2.10, Section 2.2) that we indicate with FV$_\mu$. In fact, in literature [143, 144, 208] the FV is often used considering the components associated with the mean parameters only since it results in a more compact vector representation.
- $K \in \{32, 64, 128, 256\}$: We varied the number K of Gaussian mixtures. However, we did not take into account K bigger that 256 because the dimensionality of the resulting FV would be very large and the estimation of the GMM expensive.

CNN features: We tested four different pre-trained CNN models, downloaded from the Caffe Model Zoo[3], namely

- *AlexNet (BVLC Reference CaffeNet).* This model mimics the original AlexNet [155], with minor variations as described in [146]. The model has 8 weight layers (5 convolutional + 3 fully-connected). It was trained on subset of ImageNet [88], that we refer to as ILSVRC2012[4], with about 1.5 million images and $1,000$ classes. The input image are fixed-size to 227×227 RGB.
- *OxfordNet [232].* This is an improved version of the model used by the VGG team in the ILSVRC-2014 competition. The model was trained on ILSVRC2012 dataset and contains 16 weight layers (13 convolutional + 3 fully-connected). The input image are fixed-size to 224×224 RGB.
- *PlacesNet [276].* PlaceNet model shares the same architecture of BVLC Reference CaffeNet, while being trained on 205 scene categories of Places Database [276] with about 2.4 million images.
- *HybridNet [276].* The architecture of HybridNet is the same as the BVLC Reference CaffeNet. The model was trained on 1,183 categories (205 scene categories from Places205 Database and 978 object categories from ILSVRC2012 [5]) with about 3.6 million images.

In our tests we used Caffe [146] and, for each model, we extracted the output of the last convolutional layer after pooling (*pool5*) and the first two fully-connected layers (*fc6, fc7*). The only preprocessing we did is resizing the input images to the canonical resolution (*e.g.* 227×227 pixels for the AlexNet) and then subtracting from each pixel the mean RGB value $(104, 117, 123)$ computed on ILSVRC2012 data.

Most of the papers reporting results obtained using the CNN features maintain the ReLU [63, 94, 217], *i.e.*, negative activations values are discarded replacing them with 0. In our experiments, we reported the results obtained both with and without the ReLU. Moreover, we ℓ_2-normalize the resulting descriptors, as done in [43, 63, 217].

Note that $pool5$ still contains spatial information from the input image, however it is very high dimensional ($25,088$ components for OxfordNet and $9,216$ components for AlexNet, PlacesNet, and HybridNet). $fc6$ and $fc7$ are $4,096$-dimensional vectors.

3.1.2.4 Results

We start our experimental analysis by extensively testing both the FV and CNN approaches. Then, we take into account the combination of FV and CNN features into a single image representation. Finally, we compare our best results with the retrieval performances previously achieved in [30] using BoW and VLAD image representations.

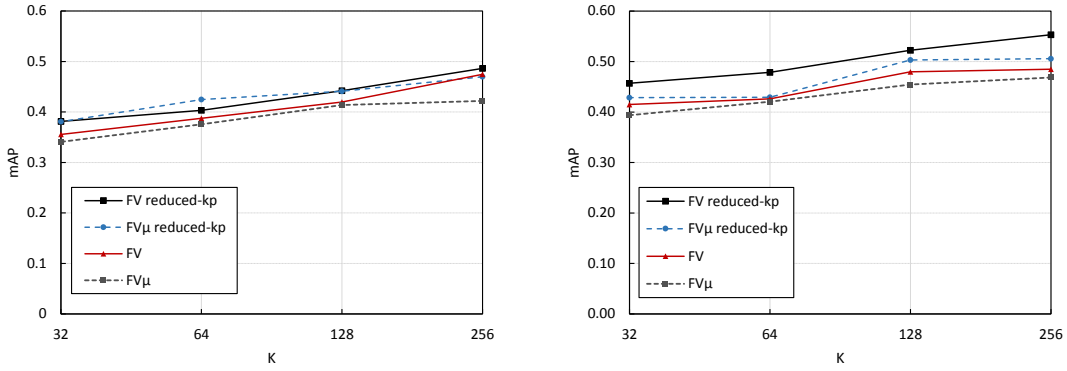
Fisher Vector In Figure 3.4 we show the mAP achieved by FV and FV$_\mu$ varying the number K of Gaussian mixtures and different local feature setting (SIFT/SIFTPCA and keypoint reduction). As expected the bigger K the better the performance. Moreover, the Fisher Vectors computed from SIFTPCA64 are more compact and more effective than the respective vectors computed from SIFT. This is a consequences of the standard FV representation that assume a diagonal covariance matrix. So, performing PCA on the local descriptors before performing FV aggregation is crucial since it decorrelated the dimensions of the descriptors (see also [144]). For this reason in the following we analyse just the results obtained using SIFTPCA64.

The whole FV performs better than FV$_\mu$ and, interestingly, the use of keypoint-reduction technique further improves the results. Thus, the overall best mAP was 0.55, obtained using FV with $K = 256$ and reduced-keypoints.

[3] https://github.com/BVLC/caffe/wiki/Model-Zoo

[4] See also Section 2.5 where all the used/cited datasets are described.

[5] Not all the 1,000 categories of ILSVRC2012 are used due to an overlap with some categories of Places205 Database.



(a) SIFT (b) SIFTPCA64

Figure 3.4: *mAP for Fisher Vector descriptors, varying the number K of mixture components of the GMM. FV indicates the full-sized Fisher Vector, while FV_μ is the Fisher Vector referred only to the mean values of the GMM. The representations are computed both using all the keypoints extracted from images and the reduced-keypoint approach. In (a) we report the results obtained using SIFT descriptors; in (b) we report the results obtained with SIFTPCA64 descriptors.*

Table 3.1: *Performance of various Fisher Vector representations. FV is the full-size Fisher Vector, while FV_μ is the Fisher Vector referred only to the mean values of the GMM. All the FVs were computed by using SIFTPCA64. The* Reduced-keypoints *column indicates if the local feature selection was used. K is the number of mixture components of the GMM.* Dims *and* bytes *are respectively the number of components and the average size in bytes of each vector representation. The results are ordered with respect to the mAP quality measure. Bold numbers denote maxima in the respective column.*

Method	**Reduced keypoints**	**K**	**Dims**	**Bytes**	**mAP**	**p** $(r = 1)$	**p** $(r = 10)$	**p** $(r = 100)$
FV	×	256	33,024	132,096	**0.55**	**0.73**	0.76	0.87
FV	×	128	16,512	66,048	0.52	0.69	0.76	0.84
FV_μ	×	256	16,384	65,536	0.51	0.69	0.77	0.86
FV_μ	×	128	8,192	32,768	0.50	0.67	0.74	0.80
FV		256	33,024	132,096	0.49	0.66	**0.79**	**0.94**
FV		128	16,512	66,048	0.48	0.64	0.76	0.93
FV	×	64	8,256	33,024	0.48	0.63	0.76	0.83
FV_μ		256	16,384	65,536	0.47	0.64	0.73	0.89
FV	×	32	4,128	16,512	0.46	0.59	0.73	0.83
FV_μ		128	8,192	32,768	0.45	0.60	0.73	0.89
FV_μ	×	32	2,048	8,192	0.43	0.57	0.66	0.79
FV_μ	×	64	4,096	16,384	0.43	0.57	0.71	0.81
FV		64	8,256	33,024	0.43	0.57	0.74	0.89
FV_μ		64	4,096	16,384	0.42	0.56	0.64	0.84
FV		32	4,128	16,512	0.42	0.59	0.67	0.83
FV_μ		32	2,048	8,192	0.39	0.59	0.67	0.79

Table 3.2: *Performance comparison of PCA-reduced FV encodings. The original FV was computed by using SIFTPCA64 with reduced-keypoints. K is the number of mixture components of the GMM.* Dims *and* bytes *are respectively the number of components and the average size in bytes of each vector representation. Bold numbers denote maxima in the respective column. The fist column is reported from Table 3.1 for reference.*

Method	Reduced keypoints	K	Dims	Bytes	mAP	p ($r = 1$)	p ($r = 10$)	p ($r = 100$)
FV	×	256	33,024	132,096	**0.55**	**0.73**	0.76	**0.87**
			PCA→ 8,192	32,768	0.54	0.71	0.74	0.86
			PCA→ 4,096	16,384	**0.55**	0.70	**0.77**	0.86
			PCA→ 2,048	8,192	0.51	0.66	0.73	0.83
			PCA→ 1,024	4,096	0.40	0.51	0.66	0.71

Table 3.1 summarizes the obtained mAP and probabilities p of finding at least one relevant image between the first r results, with $r = 1, 10, 100$. The results show that the keypoint reduction is in general useful, especially according to the mAP and $p(1)$ quality measures. For example the full-size FV with $K = 256$ and reduced-keypoints correctly recognized the query object as the first result in the 73% of cases. However, the use of all the extracted local features, respect to the reduced-keypoint approach, has led to obtain better probabilities p for bigger value of r (*e.g.* in the 94% of the cases the FV with $K = 256$ recognized the query object between the top 100 positions of the result list while the FV with $K = 256$ and reduced-keypoints reached a probability of 87%).

We already observed that for the same K the FV outperforms FV_μ and that the performances increases with increasing K. However, in order to limit the size of the Fisher Vector representation, in literature, the FV_μ have been usually preferred to the full-size FV. This is not our case, because for the same size of the final vector representation and for the same used local features, the whole FV provided us with similar performance to that of FV_μ also if the last uses bigger K. For example, FV with $K = 128$ and FV_μ with $K = 256$ have quite the same dimension (about $16,400$ components) and similar mAP (0.52/0.51) and probabilities p. However, the cost of learning the GMM and computing the FV increases with K, so in our case it would be advisable to use the whole FV (with smaller K) than FV_μ (with bigger K).

The performances obtained using 256 mixtures of Gaussian are promising, but the resulting FV is very high-dimensional. In order to reduce the cost of storing and comparing FV, we evaluated the effect of PCA-dimensionality reduction. Table 3.2 shows the results for the PCA-reduced version of the FV computed using 256 mixtures of Gaussian and reduced-keypoints. It is worth noting that the size of the FV could be effectively reduced by 88% (from $33,024$ to $4,096$ components) maintaining performance basically unchanged. Thus, it is clearly convenient to use FV in conjunction with PCA dimensionality reduction.

CNN Features In figure 3.5 and table 3.3 we report the results obtained using the outputs of the last convolutional layer (*pool5*) and the first two fully-connected layers (*fc6, fc7*) of the *OxfordNet*, *HybridNet*, *AlexNet* and *PlacesNet*. The outputs of *fc6* and *fc7* layers are considered both before and after applying the ReLU activation function.

OxfordNet exhibits the overall best performance, followed by HybridNet and AlexNet. Features extracted using PlacesNet, instead, have the lowest mAP results. Let us remind that HybridNet and PlacesNet share the same architecture of AlexNet, while being trained on different datasets: AlexNet is trained on 1.2 million images of ILSVRC2012, PlacesNet is trained on 2.4 million images of Places205 Database, and HybridNet is trained on both the previous datasets. The ILSVRC2012 is more object-centric than Places205 dataset, so it could be considered more appropriate for our test data that contains a lot of photos of peculiar objects (inscriptions) rather than scenes. Thus, our results confirm the fact, already pointed out in [63], that an appropriate choice of the training dataset may improve retrieval

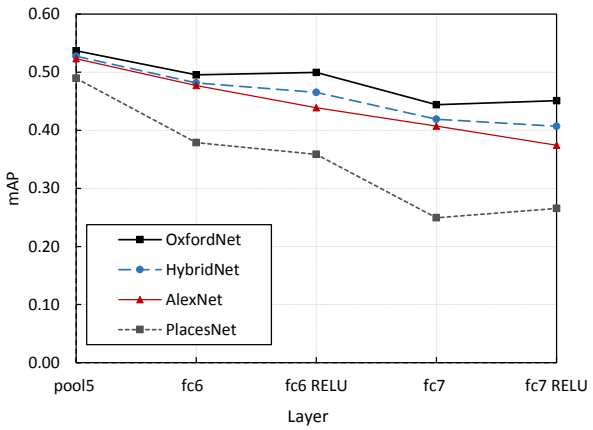


Figure 3.5: *mAP for CNN features extracted by the last convolutional layer (*pool5*) and the first two fully-connected layers (*fc6, fc7*) of some state-of-the-art pretrained models, namely* OxfordNet*,* HybridNet*,* AlexNet *and* PlacesNet*. The outputs of* fc6 *and* fc7 *layers were analysed both before and after applying the ReLU.*

performance significantly. In facts, in our case, the models trained on ILSVRC2012 perform better than the one trained on Places205 Dataset.

This suggests that results could be further improved if an epigraphic-related dataset is used for training or fine-tuning the CNN models. However, in our tests we used just pre-trained models for the following reasons. First, a large amount of *labeled* data is needed to train the networks and we had not access to such amount of epigraphic labeled images. The EDR dataset is not suitable for learning, nor for model fine-tuning because in most cases contains just one image for an inscription. Moreover, several research articles [63, 110, 217] have shown that CNN off-the-shelf features perform well for visual recognition tasks even without using fine-tuning on domain-specific dataset.

In our tests, the ReLU activation function improves the results just for the OxfordNet descriptors while the other CNNs have better performance by extracting the descriptor without applying the ReLU. Layer *pool5* performs the best for all CNNs and the performance drops with increasing in depth. We expected this result, since deep learning methods learn representations of data with multiple levels of abstraction: the higher the level, the bigger the abstraction [112]. Thus, the higher layers are not only more abstract but also more specific for the task on which it has been trained. In our case, we are interested in recognizing ancient inscriptions whose visual appearance is very different from that captured by the most of the classes of ILSVRC2012 and PlaceNet that were to train the considered CNNs model. So the lower level features as the ones extracted from *pool5* results more appropriate for our task. However, it is worth mentioning that the dimensionality of *pool5* is higher and thus the extracted feature larger.

In summary, the best results are achieved by OxfordNet *pool5* with a mAP of 0.54 and probability $p(r)$ equals to 66%, 77%, 93% respectively for $r = 1, 10, 100$. These results are similar to that of HybridNet *pool5* (mAP 0.53) and AlexNet *pool5* (mAP 0.52). However, the dimensionality of OxfordNet *pool5* is almost triple that AlexNet or HybridNet *pool5*. To overcome this issue we analysed the effect of the PCA dimensionality reduction. Table 3.4 reports results for PCA-reduced version of both OxfordNet *pool5* ad *fc6*. PCA results to be a worthwhile in term of both efficiency and effectiveness since it can considerably reduce the dimensionality of the image descriptors without loss in accuracy. Conversely, limited reduction tends to improve accuracy for both *pool5* ad *fc6*. For example, Oxford *pool5* which originally has $25,088$ components and a mAP of 0.54, reaches a mAP of 0.57 when reduced to $2,048$ components and the probabilities p are improved after the dimensionality reduction as well.

The results related to the full-dimensional image descriptors, *i.e.* without PCA dimensionality reduction, show that FV approach slightly outperforms CNN features, in fact FV with $K = 256$ reaches a mAP of 0.55 while the best mAP obtained using CNN features is 0.54 (OxfordNet *pool5*). However, the deep features used in this set of experiments are very sparse (as consequence of the ReLU activation

Table 3.3: *Performance comparison of different output layers of* OxfordNet*,* HybridNet*,* AlexNet *and* PlacesNet*.* Dims *and* bytes *are respectively the number of components and the average size in bytes of each vector representation. Results are ordered with respect to the mAP measure. Bold numbers denote maxima in the respective column.*

Method	Layer	Dims	Bytes	mAP	p ($r = 1$)	p ($r = 10$)	p ($r = 100$)
OxfordNet	pool5	25,088	100,352	**0.54**	**0.66**	0.77	**0.93**
HybridNet	pool5	9,216	36,864	0.53	**0.66**	0.81	0.90
AlexNet	pool5	9,216	36,864	0.52	**0.66**	0.81	0.89
OxfordNet	fc6 ReLU	4,096	16,384	0.50	0.64	**0.84**	0.91
OxfordNet	fc6	4,096	16,384	0.50	0.63	0.80	**0.93**
PlacesNet	pool5	9,216	36,864	0.49	0.64	0.77	0.90
HybridNet	fc6	4,096	16,384	0.48	0.59	0.80	**0.93**
AlexNet	fc6	4,096	16,384	0.48	0.59	0.83	0.87
HybridNet	fc6 ReLU	4,096	16,384	0.47	0.56	0.76	0.90
OxfordNet	fc7 ReLU	4,096	16,384	0.45	0.59	0.74	0.86
OxfordNet	fc7	4,096	16,384	0.44	0.56	0.76	0.89
AlexNet	fc6 ReLU	4,096	16,384	0.44	0.54	0.79	0.86
HybridNet	fc7	4,096	16,384	0.42	0.54	0.76	0.90
AlexNet	fc7	4,096	16,384	0.41	0.50	0.71	0.84
HybridNet	fc7 ReLU	4,096	16,384	0.41	0.53	0.71	0.87
PlacesNet	fc6	4,096	16,384	0.38	0.49	0.64	0.90
AlexNet	fc7 ReLU	4,096	16,384	0.37	0.44	0.69	0.86
PlacesNet	fc6 ReLU	4,096	16,384	0.36	0.49	0.66	0.84
PlacesNet	fc7 ReLU	4,096	16,384	0.27	0.49	0.64	0.90
PlacesNet	fc7	4,096	16,384	0.25	0.49	0.64	0.90

Table 3.4: *OxfordNet CNN: performance comparison after PCA dimensionality reduction. Results for both* pool5 *and* fc6 ReLU *are reported.* Dims *and* bytes *are respectively the number of components and the average size in bytes of each vector representation. The results related to the full-size features (*i.e. *the* pool5 *and* fc6 *features before PCA-reduction) are reported from Table 3.3 for reference. Bold numbers denote maxima in the respective column and for each approach.*

Method	Layer	Dims	Bytes	mAP	p ($r = 1$)	p ($r = 10$)	p ($r = 100$)
OxfordNet	pool5	25,088	100,352	0.54	0.66	0.77	**0.93**
		PCA→ 4,096	16,384	**0.57**	**0.70**	0.80	**0.93**
		PCA→ 2,048	8,192	**0.57**	**0.70**	**0.83**	**0.93**
		PCA→ 1,024	4,096	0.55	0.67	0.81	0.90
		PCA→ 512	2,048	0.53	0.66	0.81	0.90
		PCA→ 256	1,024	0.50	0.63	0.77	0.87
OxfordNet	fc6 ReLU	4,096	16,384	0.50	0.64	**0.84**	0.91
		PCA→ 2,048	8,192	0.52	0.66	**0.84**	0.91
		PCA→ 1,024	4,096	**0.53**	**0.67**	**0.84**	0.91
		PCA→ 512	2,048	0.52	0.66	0.83	**0.94**
		PCA→ 256	1,024	0.51	0.63	0.81	**0.94**

function) while the FV are almost dense. This explains the more robustness to dimensionality reduction showed by the CNN features, which are therefore particularly suitable when very compact image descriptors are needed to reduce the memory consumption. For example, OxfordNet *fc6* reduced to just 256-dimensional vector achieve a mAP of 0.51 that is 10% higher than the mAP achieved by FV256 reduced to the same dimension. In addition, it is interesting to note that OxfordNet *fc6* reduced to $2,048$ components (0.57 mAP) even outperform the overall best FV approach (0.55 mAP).

Combination of FV and CNN Features Here we report the results combining FV and CNN features as described in Section 3.1.1. Figure 3.6 shows the mAP obtained by combining FVs with OxfordNet *pool5* and *fc6 ReLU* features. The FVs were computed using SIFTPCA64 with reduced-keypoints and varying K from 32 to 256. The best results is a mAP of 0.75 obtained, as expected, by combining FV and CNN features with their respective best settings, *i.e.* FV with $K = 256$ (0.55 mAP) and OxfordNet *pool5* (0.54 mAP). All the combinations show an improvement with respect to the single use of CNN or FV features. It is interesting to note that, for an appropriate value of α, there is a valuable improvement also when CNN is combined with less effective Fisher Vector, such that obtained with small K. The advantage is having a minor cost for computing and storing the FV while still improving the retrieval performance. To get an idea of the costs, in our tests that use a CPU implementations, the FV encoding of a set of SIFTPCA64 descriptors required about 40 ms for $K = 64$, and about 160 ms for $K = 256$. The extraction of the CNN feature using Caffe [146] took about 300 ms per image. However, the bottleneck in the tested combinations of FV and CNN features was the SIFT extraction that took more than one second per image.

According to our results, seems that exists an optimal α for each combination of FV and CNN, *i.e.* a value such that mAP reach a maximum. The maximum performance was obtained for α between 0.2 and 0.3 when considering high number of Gaussian mixtures (*i.e.* $K = 256, 128$). Considering that the SIFT influences the overall cost of the features extraction, the proposed combinations are not suitable for applications that require high efficiency. Thus, in the following we focus on effectiveness, considering the combination of the best FV and CNN approaches, that are FV with $K = 256$ and OxfordNet *pool5* and *fc6 ReLU*. We fix $\alpha = 0.25$ since it maximizes the performance of the considered combinations. Table 3.5 summarizes the obtained results and explores also the PCA-reduced version of FV and CNN features, given that both benefit from PCA-reduction. However, for FV we did not consider reduction to less than 4,096 components since in these cases the mAP degraded as shown in Table 3.2. Interestingly, for all the combinations we recognized the query as the first result between 84% and 93% of cases and almost always we correctly recognized the query at least in the 100 top positions, even if the dimension of the original descriptors is significantly reduced.

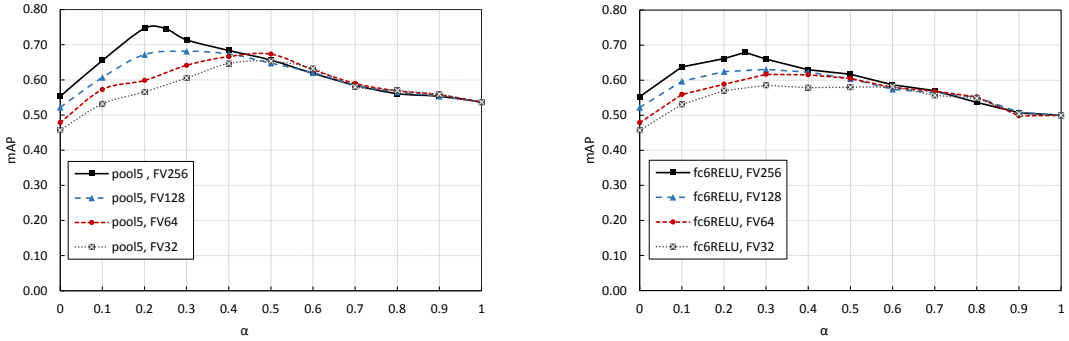
In our scenario, the combination of the OxfordNet *pool5* (reduced to 1,024 components) with the FV256 (reduced to 4,096 dimensions), could be considered as a good compromise between efficiency and effectiveness since it has a relatively small representation (5,120 components) and reaches very high performance (0.73 of mAP and $p(r)$ equals to 89%, 96%, 100% respectively for $r = 1, 10, 100$).

Comparison with the State-of-the-Art To the best of our knowledge, in literature, the topic of visual recognition of ancient inscriptions has only been faced in our previous work [30], where BoW and VLAD approaches were analysed. The experimental set up (dataset, ground truth, quality measures, local features extraction, etc.) used in [30] is the same as in this thesis, so the results are comparable.

In this section, we summarize the best results we obtained using FV, CNN and their combinations (top part of the Table 3.6) and we compare them with the results previously achieved using VLAD and BoW approaches (bottom part of Table 3.6).

In [30] the BoW approach achieved a maximum of 0.52 mAP using a visual vocabulary of $200,000$ words and performing geometric consistency check with RANSAC [102]. The VLAD approach with 256 visual words (computed using SIFT descriptors and reduced-keypoints) reached the same mAP and slight better probabilities $p(r)$. Quite worse results are obtained using BoW with 400,000 words and using the cosine similarity measure in conjunction with *tf-idf* weighting scheme.

Interestingly, all the previous results of BoW and VLAD have been overcomed by both FV and OxfordNet features, even if used individually (except for the full-size *fc6 ReLU*). For example, FV256 has quite the same dimensionality of VLAD256 but gets a 4% better mAP and slight better probabilities



(a) Combination of FVs and OxfordNet pool5 (b) Combination of FVs and OxfordNet fc6 ReLU

Figure 3.6: *mAP for various combinations of FV and OxfordNet features. $\alpha = 0$ corresponds to use only FV, while $\alpha = 1$ corresponds to use only the OxfordNet feature. The FV representations are computed varying the number K of Gaussian mixture (for $K = 32, 64, 128, 256$) and using SIFTPCA64 with reduced-keypoints. In (a) results for the combinations between FVs and OxfordNet pool5 are reported. Similarly, in (b) the results of the combinations between FVs and OxfordNet fc6 ReLU are considered.*

Table 3.5: *Performance of various mixtures of FV and OxfordNet features, by using $\alpha = 0.25$ in the convex combination of FV and CNN distances. Both full-sized and PCA-reduced features are considered. The FV representations are computed using SIFTPCA64 with reduced-keypoints.* Dims *and* bytes *columns indicate respectively the number of components and the average size in bytes of each vector representation. For each approach, the bold numbers denote maxima in the respective column.*

Method	**Dims**	**Bytes**	**mAP**	**p** $(r = 1)$	**p** $(r = 10)$	**p** $(r = 100)$
pool5, FV256	58,112	232,448	**0.75**	**0.93**	**0.97**	0.99
(pool5 →PCA 2,048), FV256	35,072	140,288	0.70	0.86	0.96	**1.00**
(pool5→PCA 1,024), FV256	34,048	136,192	0.70	0.87	0.96	**1.00**
(pool5→PCA 512), FV256	33,536	134,144	0.67	0.84	0.94	**1.00**
pool5, (FV256→PCA 4,096)	29,184	116,736	0.74	0.91	0.96	0.99
(pool5→PCA 2,048), (FV256→PCA 4,096)	6,144	24,576	0.73	0.89	0.96	**1.00**
(pool5→PCA 1,024), (FV256→PCA 4,096)	5,120	20,480	0.73	0.89	0.96	**1.00**
(pool5→PCA 512), (FV256→PCA 4,096)	4,608	18,432	0.69	0.84	0.96	**1.00**
(fc6 ReLU), FV256	37,120	148,480	**0.68**	**0.87**	0.94	**1.00**
(fc6 ReLU→PCA 2,048), FV256	35,072	140,288	**0.68**	0.86	**0.97**	**1.00**
(fc6 ReLU→PCA 1,024), FV256	34,048	136,192	**0.68**	0.86	0.94	**1.00**
(fc6 ReLU→PCA 512), FV256	33,536	134,144	**0.68**	0.86	0.93	**1.00**
(fc6 ReLU), (FV256→PCA 4,096)	8,192	32,768	0.67	0.84	0.94	**1.00**
(fc6 ReLU→PCA 2,048), (FV256→PCA 4,096)	6,144	24,576	**0.68**	0.86	**0.97**	**1.00**
(fc6 ReLU→PCA 1,024), (FV256→PCA 4,096)	5,120	20,480	**0.68**	0.84	**0.97**	**1.00**
(fc6 ReLU→PCA 512), (FV256→PCA 4,096)	4,608	18,432	**0.68**	0.84	0.94	**1.00**

Table 3.6: *Summary of the best results obtained using FV, CNN and their combinations, and comparison with the state-of-the-art results achieved using BoW and VLAD. The combinations of FV and CNN were computed using $\alpha = 0.25$. The CNN features were extracted using OxfordNet. Results related to FV and CNN are reported from Tables 3.1, 3.2, 3.3, 3.4, and 3.5. Results related to BoW and VLAD are reported from [30].* Dim *and* Bytes *are respectively the number of components and the average size in bytes of each vector representation. The results are ordered with respect to the mAP quality measure. Bold numbers denote maxima in the respective column.*

Method		**Dim**	**Bytes**	**mAP**	**p** $(r = 1)$	**p** $(r = 10)$	**p** $(r = 100)$
pool5, FV256	×	58,112	232,448	**0.75**	**0.93**	**0.97**	0.99
pool5, (FV256 →PCA 4,096)	×	29,184	116,736	0.74	0.91	0.96	0.99
(pool5→PCA 1,024), (FV256 →PCA 4,096)	×	5,120	20,480	0.73	0.89	0.96	**1.00**
(pool5→PCA 512), (FV256 →PCA 4,096)	×	4,608	18,432	0.69	0.84	0.96	**1.00**
pool5→PCA 2,048		2,048	8,192	0.57	0.70	0.83	0.93
FV256	×	33,024	132,096	0.55	0.73	0.76	0.87
FV256→PCA 4,096	×	4,096	16,384	0.55	0.70	0.77	0.86
pool5		25,088	100,352	0.54	0.66	0.77	0.93
fc6 ReLU→PCA 1,024		1,024	4,096	0.53	0.67	0.84	0.91
FV128	×	16,512	66,048	0.52	0.69	0.76	0.84
VLAD256 *[30]*	⋆	32,768	131,072	0.52	0.69	0.74	0.84
BoW 200k RANSAC *[30]*	•	4,773	19,092	0.52	0.66	0.70	0.74
BoW 400k Cos-TFIDF *[30]*	⋆	235	940	0.51	0.64	0.76	0.87

× Descriptor computed by using SIFTPCA64 with reduced-keypoints

• Descriptor computed by using SIFT

⋆ Descriptor computed by using SIFT with reduced-keypoints

$p(r)$. Furthermore, OxfordNet *pool5*, reduced to a vector of dimension $2,048$, far outperforms either BoW and VLAD both in effectiveness and memory occupation.

The overall best results were obtained by combining FV256 with OxfordNet *pool5*, with a gain over BoW and VLAD of $+22\%$ in mAP and $+24\%$ in retrieving a relevant image as the first result.

The mixture of FV256 with OxfordNet *pool5* is very high dimensional (58,112 components) so a PCA-reduced version of FV and CNN features are preferred to obtain more compact image representation while preserving effectiveness. However, as we previously observed, we did not consider reduction of FV256 to less than 4,096 components since in these cases the mAP decreases and the retrieval gain due to the features' combination do not balance the extra cost of FV256 extraction with respect to the single use of the PCA-reduced version of OxfordNet *pool5*.

In conclusion, our results show that combinations of FV and CNN achieve very high effectiveness in recognizes ancient inscription and can be profitably used when efficiency is not the main concern. The memory occupation can be downsized using PCA, and the cost of FV-CNN combination can be reduced using cheaper FV (*i.e.*, FV with small K). In Figure 3.7 we report some examples of top results obtained using different image representations. Please note that while our objective is to recognize a specific inscription (namely to have a correct answer in the first positions of the result list), the use of CNN-features (both pure and combined with FV) allow us to retrieve images that, even if are not correct answers, represent objects very similar to the query one. Thus, they are better from a qualitative point of view.

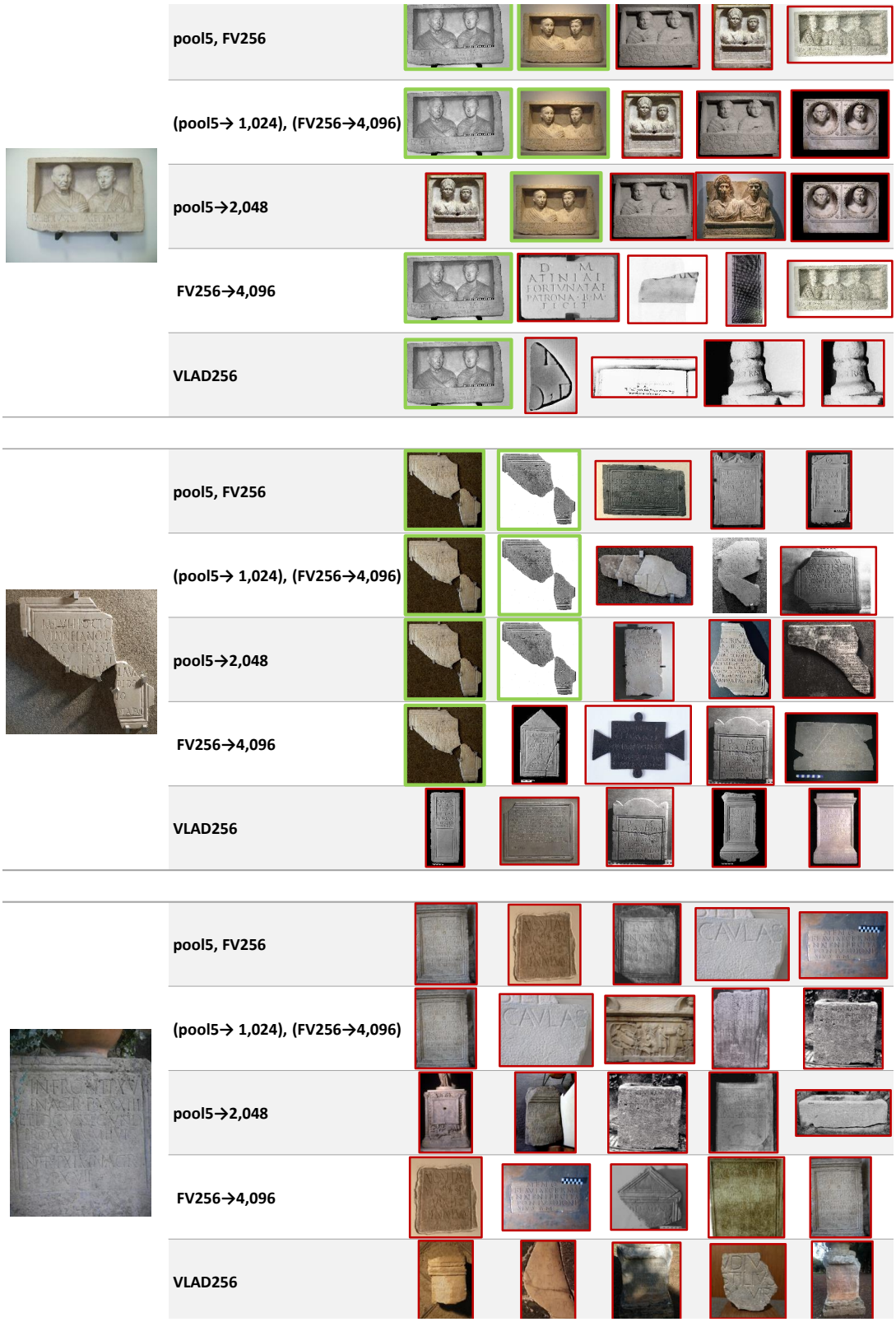


Figure 3.7: *Qualitative results for different image representation approaches. On the left, we show the query image; on the right, for each method, we report the top five results. The correct answers are outlined in green. The query images are removed from the result list.*

Table 3.7: *Performance of the combination of FV and OxfordNet* pool5 *feature, varying the parameter* α. $\alpha = 0$ *corresponds to use only FV, while* $\alpha = 1$ *corresponds to use only the OxfordNet feature. The FV was computed using* $K = 256$ *Gaussians.* dim *and* bytes *are respectively the number of components and the average size in bytes of each vector representation. Bold numbers denote maxima in the respective column.*

(a) Full-sized descriptors

Method	α	**Dim**	**Bytes**	**mAP**	**p** $(r = 1)$	**p** $(r = 10)$	**p** $(r = 100)$
FV256	0	33,024	132,096	0.56	0.85	0.97	**1.00**
pool5, FV256	0.25	58,112	232,448	**0.65**	**0.93**	**0.99**	**1.00**
pool5, FV256	0.50	58,112	232,448	0.61	0.91	0.98	**1.00**
pool5, FV256	0.75	58,112	232,448	0.58	0.89	0.98	**1.00**
pool5	1	25,088	100,352	0.56	0.88	0.97	**1.00**

(b) PCA-reduced descriptors

Method	α	**Dim**	**Bytes**	**mAP**	**p** $(r = 1)$	**p** $(r = 10)$	**p** $(r = 100)$
FV256→PCA 4,096	0	4,096	16,384	0.51	0.82	0.97	**1.00**
(pool5→PCA 1,024), (FV256→PCA 4,096)	0.25	5,120	20,480	**0.64**	**0.92**	**0.99**	**1.00**
(pool5→PCA 1,024), (FV256→PCA 4,096)	0.50	5,120	20,480	0.61	0.90	0.98	**1.00**
(pool5→PCA 1,024), (FV256→PCA 4,096)	0.75	5,120	20,480	0.58	0.88	0.98	**1.00**
pool5→PCA 1,024	1	1,024	4,096	0.56	0.87	0.98	**1.00**

3.1.3 Experiments on the Pisa Dataset

The results on the Epigraphic Database Roma reported in the previous sections show that the use of a combination of FV a CNN features leads to improve the retrieval performance with respect to use the FV or the CNN feature alone. In this Section, we further analyse the retrieval performance of FV-CNN combination in another cultural heritage context. To this scope, we used the publicly available *Pisa Dataset* [10] (described in Section 2.5.1) that contains photos of 12 monuments and landmarks located in Pisa. This dataset has also been used in [26,45,150] for classification and indexing tasks. The experimental settings and the performance measure are the same used in the previous section. The total number of queries is 921.

In Table 3.7 (a) we report the results obtained using FV with $K = 256$, OxfordNet *pool5* and their combinations for various values of the parameter α (used in the convex combination). Also in this scenario the combination of FV and CNN features leads to improve the retrieval performance. As happened in the experiments on the Epigraphic Database Rome, we observed that exists an optimal value for the parameter α where the combination achieved the maximum mAP. The optimal α was obtained around 0.25, where the corresponding mAP of 0.65 is achieved, while the use of either FV or CNN leads to have a mAP of 0.56. For $\alpha = 0.25$ we also correctly recognize the query as first results in the 93% of the cases.

In Table 3.7 (b) we evaluated the case in which the PCA is used to reduce the dimensionality of FV and CNN feature before the combination. We reduced the OxfordNet *pool5* from 25,088 to 1,024 components and the FV representation from $33,024$ to $4,096$. Also in this case the optimal α was 0.25. The retrieval results obtained using the PCA-reduced version of FV and CNN feature are in line with that obtained using the full-sized descriptors. Thus also in this case it is better to use the PCA reduction to have more compact representations while preserving the effectiveness.

3.1.4 Summary

The previous sub-sections investigated the problem of visually recognize ancient inscriptions, or other objects of cultural heritage, by testing state-of-the-art approaches for instance retrieval (VLAD, BoW, FV, CNN features).

Extensive experiments showed that very high effectiveness can be achieved by combining FV and CNN features. In fact, in more than 90% of the cases we correctly recognized the query objects using a 1-NN classifier. Moreover, we achieved a mean average precision near to 0.70, which means that the overall ordering of the results is good. Nevertheless, the combination of FV and CNN is costly due to the extraction and storage of both FV and CNN features. We showed that PCA dimensionality reduction can be effectively used to reduce the memory occupation of FV-CNN combination without loss in accuracy. In addition, the cost of the FV computation can be reduced using small number of Gaussian mixtures since also in this case we observed that the retrieval performance of CNN features is improved by the combination with the FV. However, the extraction of both the image descriptors may still be too costly to be dealt with devices with very limited resources, especially due to the SIFTs extraction. An efficient alternative, proposed in the next section, is combining CNN features with a FV encoding of binary local features, whose extraction process is up two orders of magnitude faster than SIFT.

3.2 BMM-FV: A Novel Efficient Approach for Encoding Binary Features

Now we focus on cases where the efficiency is one of the main concerns, as when considering applications running on devices with low resources, or when response time must be very fast. For this reason, we consider the case where the *binary* local features are extracted and used to represent images, since their extraction process is notable efficient [126, 221]. These features have a compact binary representation that is not the result of a quantization, but rather is computed directly from pixel-intensity comparisons (see Section 2.2.1).

Even if the process for extracting binary features is very efficient, each image is still represented by thousands of local descriptors and, so there is a significant amount of memory consumption and time required to compare local features within large databases. While the information provided by each local feature is crucial for tasks such as image stitching and 3D reconstruction, for other tasks, including image retrieval, the aggregation techniques provide useful summaries of the local features of an image. The advantage is twofold: 1) reduction of the cost of image comparison (each image is represented by a single descriptor rather than thousands of descriptors); 2) aggregated descriptors have been proved to be particularly effective for image retrieval and classification tasks. Nevertheless, aggregation methods are defined and used almost exclusively in conjunction with non-binary features, so the cost of extracting local descriptors and to aggregate them on the fly is still high.

In this section, we investigate the use of state-of-the-art aggregation methods on the top of binary local features. The objective is to improve efficiency and reduce computing resources needed for image matching by leveraging on

We expect this topic to be relevant for applications running on devices with low CPU and memory resources, as for instance mobile and wearable devices, that use binary features. In these cases, the combination of aggregation methods with binary local features leads to scale up image search on large scale, where direct matching is not feasible. In this section we:

- provide an extensive comparison and analysis of the aggregation methods applied to binary local descriptors on two standard benchmarks, namely INRIA Holidays [139] and Oxford5k [211];
- propose a novel formulation of the Fisher Vector to encode binary descriptors, named BMM-FV. Our FV encoding is built using a Bernoulli Mixture Model (BMM) and preserves the structure of the traditional FV built using a Gaussian Mixture Model, so existing implementations of the FV can be easily adapted to work also with BMMs;
- evaluate our BMM-FV on the top of several binary local features (ORB [221], LATCH [166], A-KAZE [17]) whose performances have not been yet reported on benchmark for image retrieval;

- evaluate the combination of the BMM-FV (and other encodings of binary features) with the emerging CNN features, including experiments on a large scale.

Our results show that the use of aggregation methods with binary local descriptors is generally effective even if, as expected, retrieval performance is worse than that obtained applying the same methods directly to non-binary features. The BMM-FV approach provided us with performance results that are better than all the other tested aggregations of binary descriptors. In addition, we found that some aggregation methods lead to obtain very compact image representation with a retrieval performance comparable to that of direct matching, which actually is the most used approach to evaluate the similarity of images described by binary local features. Finally, we show that the combinations of BMM-FV and CNN improve the latter retrieval performances and achieves effectiveness in line with that obtained combining CNN and FV built upon the costly SIFTs.

In the following sections we first review how traditional BoW and VLAD methods have been adapted in literature to work with binary features (Section 3.2.1). Then we present our BMM-FV encoding (Section 3.2.2), and finally, we report the experimental analysis (Section 3.2.3).

3.2.1 BoW and VLAD Encodings of Binary Local Features

Traditional BoW and VLAD approaches use a "visual vocabulary" to encode the local descriptors of an image into a single fixed-size descriptor (see Sections 2.2.2.1 and 2.2.2.2). The visual vocabulary is built by clustering a large set of local descriptors, where the centroids act as "visual words". The *k-means* [171] clustering is traditionally used to this scope.

In order to extend these encoding schemes to deal with binary features, we need a cluster algorithm able to cope with binary strings and with the Hamming distance. The *k-medoids* [151] algorithms is suitable for this scope, but it requires a computational effort to calculate a full distance matrix between the elements of each cluster. In [114] it was proposed to use a voting scheme, named *k-majority*, to process a collection of binary vectors and seek for a good set of centroids to be used with the BoW model. Initially the centroids are randomly selected and each element of the collection is associated with its nearest centroid (according to the Hamming distance). Subsequently, for each cluster, a new centroid is computed by assigning 1 to its i-th bit if the majority of the binary vectors in the cluster have a 1 in the i-th bit. The algorithm iterates until no centroids are changed during the previous iteration. After the visual words are computed, the BoW aggregation is performed as usual, but using the Hamming distance rather than the Euclidean when searching the closest centroid to a given descriptor. In [163,274] the BoW model and the k-means clustering were modified to fit the binary features by replacing the Euclidean distance with the Hamming distance, and by replacing the mean operation with the median operation. It is immediate to prove that the resulting representation is equivalent to the BoW based on k-majority.

Of course, even in not explicitly investigated in literature, the k-majority and the k-medoids can be also used to build the visual vocabulary for the VLAD representations. Alternatively, a naive way to apply the VLAD scheme to binary local descriptors is treating binary vectors as a particular case of real-valued vectors. In this way, the k-means algorithm can be used to build the visual vocabulary and the difference between the centroids and the descriptors can be accumulated as usual. This approach was used in [257], where a variation to the VLAD image signature, called BVLAD, was defined to work with binary features. Specifically, the BVLAD is the binarization of a VLAD obtained using power-law, intra-normalization, ℓ_2 normalization and multiple PCA. Thereafter we have not evaluated the performance of the BVLAD because, unlike the other techniques of aggregation of binary features, it uses binarization by thresholding which is an extra step after the aggregation phase. The binarization of the final image signature was out of the scope our study.

3.2.2 BMM-FV and Other FV Encodings of Binary Features

The main idea of the Fisher Vector encoding (see Section 2.2.2.3) is to represents a set of data objects, like the set of local descriptors extracted from an image, in term of its statistically property with respect to a probability distribution $p(\cdot|\lambda)$ of some parameters $\lambda \in \mathbb{R}^m$. The distribution $p(\cdot|\lambda)$ models a generative process in the space of all the possible data observations, and typically it is estimated on a training set.

Traditional definition of FV, uses a GMM to model the data, considering that any continuous distribution can be approximated arbitrarily well by an appropriate finite Gaussian mixture [180, ch.6].

Since the Bernoulli distribution models an experiment that has only two possible outcomes (0 and 1), a reasonable alternative to characterize the distribution of a set of binary descriptors is to use Bernoulli Mixture Model (BMM). In order to encode binary local features, we derive and test an extension of the FV, which we name *BMM-FV*, built using a BMM . Specifically, we chose $p(\cdot|\lambda)$ to be multivariate Bernoulli mixture with K components and parameters $\lambda = \{w_k, \boldsymbol{\mu}_k = [\mu_{k1}, \ldots, \mu_{kD}]\}_{k=1,\ldots,K}$:

$$p(\mathbf{x}_t|\lambda) = \sum_{k=1}^{K} w_k p_k(\mathbf{x}_t), \qquad \mathbf{x}_t \in \{0,1\}^D \tag{3.2}$$

where

$$p_k(\mathbf{x}_t) = \prod_{d=1}^{D} \mu_{kd}^{x_{td}} (1-\mu_{kd})^{1-x_{td}} \tag{3.3}$$

and

$$\sum_{k=1}^{K} w_k = 1, \quad w_k > 0 \qquad \forall\, k = 1, \ldots, K. \tag{3.4}$$

To avoid enforcing explicitly the constraints in (3.4), we used the soft-max formalism [154, 226] for the weight parameters:

$$w_k = \exp(\alpha_k) / \sum_{i=1}^{K} \exp(\alpha_i). \tag{3.5}$$

So, given a set $X = \{\mathbf{x}_t,\, t = 1, \ldots, T\}$ of D-dimensional binary vectors, $\mathbf{x}_t \in \{0,1\}^D$, and assuming that the samples are independent we have that the score vector $\boldsymbol{G}_\lambda^X$ with respect to the parameter $\lambda = \{\alpha_k, \boldsymbol{\mu}_k\}_{k=1,\ldots,K}$ is (see Appendix A.1) the concatenation of

$$G_{\alpha_k}^X = \sum_{t=1}^{T} \frac{\partial \log p(\mathbf{x}_t|\lambda)}{\partial \alpha_k} = \sum_{t=1}^{T} (\gamma_t(k) - w_k)$$

$$G_{\mu_{kd}}^X = \sum_{t=1}^{T} \frac{\partial \log p(\mathbf{x}_t|\lambda)}{\partial \mu_{kd}} = \sum_{t=1}^{T} \gamma_t(k) \left(\frac{x_{td} - \mu_{kd}}{\mu_{kd}(1-\mu_{kd})} \right)$$

where $\gamma_t(k) = p(k|\mathbf{x}_t, \lambda)$ is the *occupancy probability* (or posterior probability). The occupancy probability $\gamma_t(k)$ represents the probability for the observation $\mathbf{x}_t$ to be generated by the k-th Bernoulli and it is calculated as $\gamma_t(k) = w_k p_k(\mathbf{x}_t) / \sum_{j=1}^{K} w_j p_j(\mathbf{x}_t)$.

The FV of a set X is then obtained by normalizing the score $\boldsymbol{G}_\lambda^X$ by the matrix L_λ, which is the square root of the inverse of the Fisher Information Matrix (FIM), and by the sample size T. In the Appendix A.2 we provide an approximation of FIM under the assumption that the occupancy probability $\gamma_t(k)$ is sharply peaked on a single value of k for each descriptor $\mathbf{x}_t$, obtained following an approach very similar to that used in [226] for the GMM case. By using our FIM approximation, we got the following normalized gradient:

$$\mathcal{G}_{\alpha_k}^X = \frac{1}{T\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k)$$

$$\mathcal{G}_{\mu_{kd}}^X = \frac{1}{T\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{x_{td} - \mu_{kd}}{\sqrt{\mu_{kd}(1-\mu_{kd})}} \right)$$

The final BMM-FV is the concatenation of $\mathcal{G}_{\alpha_k}^X$ and $\mathcal{G}_{\mu_{kd}}^X$ for $k = 1, \ldots, K$, $d = 1, \ldots, D$ and is therefore of dimension $K(D+1)$.

Table 3.8: *Comparison of the structure of the FVs derived using BMM with that derived using GMM. Parameters for BMM are $\lambda^B = \{w_k^B, \boldsymbol{\mu}_k^B\}_{k=1,\dots,K}$ and for GMM are $\lambda^G = \{w_k^G, \boldsymbol{\mu}_k^G, \boldsymbol{\Sigma}_k^G\}_{k=1,\dots,K}$, where $\boldsymbol{w}_k^B$, $\boldsymbol{\mu}_k^B$ are the mixture weight and the mean vector of the k-th Bernoulli and $\boldsymbol{w}_k^G$ $\boldsymbol{\mu}_k^G$, $\boldsymbol{\Sigma}_k^G$ are respectively the mixture weight, mean vector and the (diagonal) covariance matrix of the k-th Gaussian.*

Traditional FV (GMM-FV) [226]

$$\mathcal{G}^X_{\alpha_k^G} = \frac{1}{T\sqrt{w_k^G}} \sum_{t=1}^{T} \left(\gamma_t^G(k) - w_k^G\right)$$

$$\mathcal{G}^X_{\mu_{kd}^G} = \frac{1}{T\sqrt{w_k^G}} \sum_{t=1}^{T} \gamma_t^G(k) \frac{x_{td} - \mu_{kd}^G}{\sigma_{kd}^G}$$

$$\mathcal{G}^X_{\sigma_{kd}^G} = \frac{1}{T\sqrt{w_k^G}} \sum_{t=1}^{T} \gamma_t^G(k) \frac{1}{\sqrt{2}} \left[\frac{(x_{td} - \mu_{kd}^G)^2}{(\sigma_{kd}^G)^2} - 1\right]$$

BMM-FV (our formalization)

$$\mathcal{G}^X_{\alpha_k^B} = \frac{1}{T\sqrt{w_k^B}} \sum_{t=1}^{T} \left(\gamma_t^B(k) - w_k^B\right)$$

$$\mathcal{G}^X_{\mu_{kd}^B} = \frac{1}{T\sqrt{w_k^B}} \sum_{t=1}^{T} \gamma_t^B(k) \frac{x_{td} - \mu_{kd}^B}{\sqrt{\mu_{kd}^B(1-\mu_{kd}^B)}}$$

BMM-FV (Uchida et. al [251])

($\mathcal{G}^X_{\alpha_k^B}$ not explicitly derived in [251])

$$\mathcal{G}^X_{\mu_{kd}^B} = \frac{\sum_{t=1}^{T} \gamma_t(k) \frac{(-1)^{1-x_{td}}}{(\mu_{kd}^B)^{x_{td}}\left(1-\mu_{kd}^B\right)^{1-x_{td}}}}{T\sqrt{T w_k^B \left(\frac{\sum_{i=1}^{K} w_i^B \mu_{id}^B}{(\mu_{kd}^B)^2} + \frac{\sum_{i=1}^{K} w_i^B (1-\mu_{id}^B)}{(1-\mu_{kd}^B)^2}\right)}}$$

Sànchez [226] highlights that the FV derived from GMM can be computed in terms of the following 0-order and 1-order statistics: $S_k^0 = \sum_{t=1}^{T} \gamma_t(k) \in \mathbb{R}$, $\boldsymbol{S}_k^1 = \sum_{t=1}^{T} \gamma_t(k)\mathbf{x}_t \in \mathbb{R}^D$. Our BMM-FV can be also written in terms of these statistics as

$$\mathcal{G}^X_{\alpha_k} = \frac{1}{T\sqrt{w_k}}(S_k^0 - T w_k)$$

$$\mathcal{G}^X_{\mu_{kd}} = \frac{S_{kd}^1 - \mu_{kd} S_k^0}{T\sqrt{w_k \mu_{kd}(1-\mu_{kd})}}.$$

We finally observe that, as done with the traditional FV encoding, we used power-law and ℓ_2 normalization of the final BMM-FV to improve its effectiveness.

An extension of the FV by using the BMM has been also carried in [227, 251]. Our approach differs from the one proposed in [251] in the approximation of the square root of the inverse of the FIM (*i.e.*, $\boldsymbol{L}_\lambda$) . It is worth noting that our formalization preserves the structure of the traditional FV derived by using the GMM, where Gaussian means and variances are replaced by Bernoulli means μ_{kd} and variances $\mu_{kd}(1-\mu_{kd})$ (see Table 3.8). In [227], the FV formalism was generalized to a broader family of distributions knows as the *exponential family* that encompasses the Bernoulli distribution as well as the Gaussian one. However, [227] lacks in an explicit definition of the FV and of the FIM approximation in the case of BMM which was out of the scope of their work. Our formulation differs from that of [227] in the choice of the parameters used in the gradient computation of the score function [6]. A similar difference holds also for the FV computed in [227] using a GMM, given that in [227] the score function is computed

[6] A Bernoulli distribution $p(x) = \mu^x(1-\mu)^{1-x}$ of parameter μ can be written as exponential distribution $p(x) = exp(\eta x -$

w.r.t. the natural parameters of the Gaussian distribution rather than the mean and the variance parameters which are typically used in literature for the FV representation [206,208,226]. Unfortunately, the authors of [227] did not experimentally compare the FVs obtained using or not the natural parameters.

In our experiments, the FV derived as in [251] obtained very similar retrieval performance to that of our BMM-FV, thus we report just the results obtained by using our formulation. Furthermore, we have not experimentally evaluated the FVs computed using the gradient with respect to the natural parameters of a BMM or a GMM as described in [227], because the evaluation of the performance obtained using or not the natural parameters in the derivation of the score function is a more general topic which reserve to be further investigated outside the specific context of the encodings binary local features.

3.2.3 Experiments

We now evaluate and compare the performance of aggregations of binary local features on two benchmarks for image retrieval. We also compare the results against matching the local features without any aggregation, which actually is the most used approach whenever the images are represented by binary local features. First, we introduce the experimental settings (Section 3.2.3.1). Then we compare BoW, VLAD, FV based on the GMM, and BMM-FV approaches to aggregate ORB binary features (Section 3.2.3.2). The performance of the BMM-FVs is evaluated using different binary features (namely ORB, LATCH, and A-KAZE). We further test the combination of our BMM-FV with the CNN features (Section 3.2.3.3). Finally, we report experimental results on large scale (Section 3.2.3.4).

3.2.3.1 Experimental Setup

Below is the description on the datasets, the performance measure and feature extraction procedures used in the experiments.

Datasets The experiments were conducted using two publicly available datasets, namely *INRIA Holidays* [139] and *Oxford5k* [211], that are de-facto standard benchmarking datasets for image retrieval [40,139,143,144,208,245,275]. As in many other articles, *e.g.* [139,143,144,212], all the learning stages (clustering, etc.) were performed off-line using independent image collections. *Flickr60k* dataset [139] was used as training set for INRIA Holidays, while *Paris6k* dataset [212] was used as training set for Oxford5k.

For large-scale experiments we combined the Holidays dataset with the 1 million *MIRFlickr* dataset [135], used as distractor set as in [23, 139]. Compared to Holidays, the Flickr datasets is slightly biased, because it includes low-resolution images and more photos of humans.

We described all these datasets in Section 2.5. Here we additionally note that when using INRIA Holidays, as done by the authors of the dataset, we resized the images to a maximum of $786,432$ pixels (768 pixels for the smaller dimension) before extracting the local features.

Features Comparison and Performance Measure The cosine similarity in conjunction with a term weighting scheme (*e.g.*, tf-idf) is adopted for evaluating the similarity between BoW representations, while the Euclidean distance is used to compare VLAD, FV and CNN-based image signatures. Note that in our case the Euclidean distance is equivalent to the cosine similarity ($s_{\cos}(x_1, x_2) = (x_1 \cdot x_2)/(\|x_1\|_2\|x_2\|_2)$) since the vectors are ℓ_2-normalized [7].

The image comparison based on the *direct matching* of the local features (*i.e.* without aggregation) was performed adopting the distance ratio criterion proposed in [126, 173]. Specifically, candidate matches to local features of the image query are identified by finding their nearest neighbors in the

$log(1 + e^{\eta}))$ where $\eta = \log(\frac{\mu}{1-\mu})$ is the *natural parameter*. In [227] the score function is computed considering the gradient w.r.t. the *natural parameters* η while in this paper we used the gradient w.r.t. the standard parameter μ of the Bernoulli (as also done in [251]).

[7] If the vectors are ℓ_2- normalized then searching for the objects with greatest cosine similarity to the query is equivalent to searching for the objects with lowest Euclidean distance from the query. In fact, in such case, we have $s_{\cos}(x_1, x_2) = 1 - \frac{1}{2}\ell_2(x_1, x_2)^2$, which implies that the ranked list of the results to a query is the same (*i.e.*, $\ell_2(x_1, x_2) \leq \ell_2(x_1, x_3)$ if, and only if, $s_{\cos}(x_1, x_2) \geq s_{\cos}(x_1, x_3) \,\forall\, x_1, x_2, x_3$).

database of images. Matches are discarded if the ratio of the distances between the two closest neighbours is above the 0.8 threshold. The similarity between two images is computed as the percentage of matching pairs with respect to the total local features in the query image.

The retrieval performance of each method was measured by the mAP. In the experiments on INRIA Holidays, we computed the average precision after removing the query image from the ranking list. In the experiments on Oxford5k, we removed the *junk* images from the ranking before computing the average precision, as recommended in [211] and in the evaluation package provided with the dataset.

Features Extraction Details on how the features for the various approaches were extracted are reported in the following.

Local features: In the experiments we used ORB [221], LATCH [166], and A-KAZE [17] binary local features. We used OpenCV [56] to extract up to $2,000$ local features per image.

Visual Vocabularies and Mixture Models: The visual vocabularies used for building the BoW and VLAD representations were computed using several clustering algorithms, *i.e.* k-medoids, k-majority and k-means. The k-means algorithm was applied to the binary features by treating the binary vectors as real-valued vectors.

The parameters λ^B of the BMM and λ^G of the GMM were learned independently by using the the Expectation Maximization (EM) algorithm [50, ch.9]. As stopping criterion, we used the convergence in ℓ_2-norm of the mean parameters, choosing $\epsilon = 0.05$ as convergence threshold. As suggested in [50, ch.9], the BMM/GMM parameters used in EM algorithm were initialized with: (a) $1/K$ for the mixing coefficients w_k^B and w_k^G; (b) random values chosen uniformly in the range $(0.25, 0.75)$, for the BMM means; (c) centroids precomputed using k-means for the GMM means; (d) mean variance of the clusters found using k-means for the diagonal elements of the GMM covariance matrices.

All the learning stages, *i.e.* k-means, k-medoids, k-majority and the estimation of GMM/BMM, were performed using in order of 1M descriptors randomly selected from the local features extracted from the training sets (namely Flickr60k for INRIA Holidays and Paris6k for Oxford5k).

BoW, VLAD, FV: The various encodings of the local features (as well as the visual vocabularies and the mixture models) were computed using VIR [98], where we integrated the code for computing the BMM-FV. These representations are all parametrized by a single integer K. It corresponds to the number of centroids (visual words) used in BoW and VLAD, and to the number of mixture components of GMM/BMM used in FV representations.

For the FVs, we used only the components $\mathcal{G}_\mu$ associated with the mean vectors because, in this context, we observed that the components related to the mixture weights do not improve the results much.

As a common post-processing step [144, 209], both the FVs and the VLADs were power-law normalized and subsequently ℓ_2-normalized. The exponent $\beta = 0.5$ was used in the power-law normalization. We also applied PCA to reduce VLAD and FV dimensionality. The projection matrices were estimated on the training datasets.

CNN features: We used Caffe [146] on the pre-trained HybridNet [276] model to extracted the output of the first fully-connected layer (*fc6*) after applying the ReLU. The resulting $4,096$-dimensional descriptors were ℓ_2 normalized. As preprocessing step we warped the input images to the canonical resolution of 227×227 RGB and we subtract (from each pixel) the global mean RGB value $(104, 117, 123)$ computed on ILSVRC2012.

3.2.3.2 Comparison of Various Encodings of Binary Local Features

In Table 3.9 we summarize the retrieval performance of various aggregation methods applied to ORB features, namely the BoW, the VLAD, the FV based on the GMM, and the BMM-FV. In addition, in the last line of the table, we report the results obtained without any aggregation, which we refer to as the *direct matching* of local features.

Table 3.9: *Performance evaluation of various aggregation methods applied on ORB binary features. K indicates the number of centroids (visual words) used in BoW and VLAD and the number of mixture components of GMM or BMM used in FV representations;* Dims *is the number of components of each vector representation. Bold numbers denote maxima in the respective column.*

Method	**Local Feature**	**Learning Method**	**K**	**Dims**	**mAP (%)**	
					Holidays	Oxford5k
BoW	ORB	k-means	20,000	20,000	44.9	22.2
BoW	ORB	k-majority	20,000	20,000	44.2	22.8
BoW	ORB	k-medoids	20,000	20,000	37.9	18.8
VLAD	ORB	k-means	64	16,384	47.8	23.6
VLAD	ORB	k-means	64	PCA→ 1,024	46.0	23.2
VLAD	ORB	k-means	64	PCA→ 128	30.9	19.3
VLAD	ORB	k-majority	64	16,384	32.4	16.6
VLAD	ORB	k-medoids	64	16,384	30.6	15.6
FV	ORB	GMM	64	16,384	42.0	20.4
FV	ORB	GMM	64	PCA→ 1,024	42.6	20.3
FV	ORB	GMM	64	PCA→ 128	35.5	19.6
FV	ORB	BMM	64	16,384	49.6	**24.3**
FV	ORB	BMM	64	PCA→ 1,024	**51.3**	23.4
FV	ORB	BMM	64	PCA→ 128	44.6	19.1
Direct matching	ORB				38.1	31.7

Table 3.10: *Retrieval performance of our BMM-FV on INRIA Holidays and Oxford5k. K is the number of BMM mixtures.* Dims *is the number of components of the final vector representation. Bold numbers denote maxima in the respective column.*

(a) *Performance evaluation for increasing number K of Bernullian mixture components*

K	**Dims**	**mAP** (%)	
		Holidays	Oxford5k
4	1,024	32.0	14.3
8	2,048	38.2	17.4
16	4,096	41.9	19.4
32	8,192	45.9	21.3
64	16,384	49.6	24.3
128	32,768	52.3	26.4
256	65,536	53.0	27.3
512	131,072	**54.7**	**27.4**

(b) *Results after dimensionality reduction when $K = 64$ Bernoulli are used*

K	**Dims**	**mAP** (%)	
		Holidays	Oxford5k
64	16,384	49.6	24.3
64	PCA→ 4,096	**52.6**	**25.1**
64	PCA→ 2,048	51.8	24.3
64	PCA→ 1,024	51.3	23.4
64	PCA→ 512	48.2	21.7
64	PCA→ 256	45.9	20.3
64	PCA→ 128	44.6	19.1
64	PCA→ 64	42.9	17.2

Table 3.11: *Baseline aggregation methods on non-binary local features. Results are reported from [143, 144] for reference.*

Method	Local Feature	Learning Method	K	Dims	mAP (%)	
					Holidays	Oxford5k
BoW	SIFT	k-means	20,000	20,000	40.4	-
BoW	SIFT PCA 64	k-means	20,000	20,000	43.7	35.4
VLAD	SIFT	k-means	64	8,192	52.6	-
VLAD	SIFT	k-means	64	PCA$\rightarrow$ 128	51.0	-
VLAD	SIFT PCA 64	k-means	64	4,096	55.6	37.8
VLAD	SIFT PCA 64	k-means	64	PCA$\rightarrow$ 128	55.7	28.7
FV	SIFT	GMM	64	8,192	49.5	-
FV	SIFT	GMM	64	PCA$\rightarrow$ 128	49.2	-
FV	SIFT PCA 64	GMM	64	4,096	59.5	41.8
FV	SIFT PCA 64	GMM	64	PCA$\rightarrow$ 128	56.5	30.1

Among the various baseline aggregation methods (*i.e.* without using PCA), the BMM-FV approach achieves the best retrieval performance, that is a mAP of 49.6% on Holidays and 24.3% on Oxford. PCA dimensionality reduction from $16,384$ to $1,024$ components, applied on BMM-FV, marginally reduces the mAP on Oxford5k, while on Holiday allows us to get 51.3% that is, for this dataset, the best result achieved between all the other aggregation techniques tested on ORB binary features.

Good results are also achieved using VLAD in conjunction with k-means, which obtains a mAP of 47.8% on Holidays and 23.6% on Oxford5k.

The BOW representation allows to get a mAP of 44.9%, 44.2%, 37.9% on Holidays and 22.2%, 22.8%, 18.8% on Oxford5k using respectively k-means, k-majority, and k-medoids for the learning of a visual vocabulary of $20,000$ visual words.

The GMM-FV method gives results slight worse than BoW: 42.0% of mAP on Holidays and 20.4% of mAP on Oxford5k. The use of PCA to reduce dimensions from $16,384$ to $1,024$ lefts the results of GMM-FV on Oxford5k substantially unchanged while slightly improved the mAP on Holidays (42.6%).

Finally, the worst performance is that of VLAD in combination with vocabularies learned by k-majority (32.4% on Holidays and 16.6% on Oxford) and k-medoids (30.6% on Holidays and 15.6% on Oxford).

It is generally interesting to note that on INRIA Holidays, the VLAD with k-means, the BoW with k-means/k-majority, and the FVs are better than direct match. In fact, mAP of direct matching of ORB descriptors is 38.1% while on Oxford5k the direct matching reached a mAP of 31.7%.

In Table 3.10 we also report the performance of our derivation of the BMM-FV varying the number K of Bernoulli mixture components and investigating the impact of the PCA dimensionality reduction in the case of $K = 64$. In Table (3.10a) we can see that on the Holidays dataset, the mAP grows from 32.0% when using only 4 mixtures to 54.7% when using $K = 512$. On Oxford5k, mAP varies from 14.3% to 27.4%, respectively, for $K = 4$ and $K = 512$. Table (3.10b) shows that the best results for $K = 64$ are achieved when reducing the full size BMM-FV to $4,096$ with a mAP of 52.6% for Holidays and 25.1% for Oxfrod5k.

For completeness, in Table 3.11, we also report the results of the same baseline encodings approaches applied to non-binary features (both full-size SIFT and PCA-reduced to 64 components) taken from literature [143,144]. As expected, aggregation methods in general exhibit better performance in combination with SIFT/SIFTPCA then with ORB, especially for the Oxford5k dataset. However, it is worth noting that on the INRIA Holidays the BMM-FV outperforms the BoW on SIFT/SIFTPCA and reach a similar performance of the FV built upon SIFTs.

Summing up, the results show that in the context of binary local features the BMM-FV outperforms the compared aggregation methods, namely the BoW, the VLAD and the GMM-FV. The performance of the BMM-FV is an increasing function of the number K of Bernoulli mixtures. However, for large K,

Table 3.12: *Average time costs for computing various image representations, using a CPU implementation, on an Intel i7 3.5 GHz. The cost of computing the CNN feature of an image was estimated using a pretrained AlexNet-like model and the Caffe framework [146]. The values related to the FV refers only to the cost of aggregating the local descriptors of an image into a single vector and do not encompass the cost of extracting the local features, neither the learning of the Gaussian or the Bernoulli Mixture Model which is calculated off-line. The cost of computing FV varies proportionally with the quantity TKD, where T is the number of local features extracted from an image, K is the number of mixtures of Gaussian/Bernoulli, and D is the dimensionality of each local feature; we reported the approximate cost for $T = 2,000$ and $KD = 64 * 64$ and $KD = 64 * 256$. The cost of SIFT/ORB local feature extraction was estimated according to [126] by considering about $2,000$ features per image.*

	CNN	FV Encoding	SIFT	ORB
Computing time per image	$\sim$300 ms	$\sim$40 ms [$KD = 64 * 64$] $\sim$160 ms [$KD = 256 * 64$]	$\sim$1200 ms	$\sim$26 ms

the improvement tends to be smaller and the dimensionality of the FV becomes very large (*e.g.* $65,536$ dimensions using $K = 256$). Hence, for high values of K, the benefit of the improved accuracy is not worth the computational overhead (both for the BMM estimation and for the cost of storage/comparison of FVs).

The PCA reduction of BMM-FV is effective since it can provide a very compact image signature with just a slight loss in accuracy, as shown in the case of $K = 64$ (Table 3.10b). Dimension reduction does not necessarily reduce the accuracy. Conversely, limited reduction tends to improve the retrieval performance of the FV representations.

For the computation of VLAD, the k-means results are more effective than k-majority/k-medoids clustering, since the use of non-binary centroids gives more discriminant information during the computation of the residual vectors used in VLAD.

For the BoW approach, k-means and k-majority performs equally better than k-medoids. However, the k-majority is preferable in this case because the cost of the quantization process is significantly reduced by using the Hamming distance, rather than Euclidean one, when comparing centroids and binary local features.

Both BMM-FV and VLAD, with only $K = 64$, outperform BoW. However, as happens for non-binary features (see Table 3.11), the loss in accuracy of BoW representation is comparatively lower when the variability of the images is limited, as for the Oxford5k dataset. As expected, BMM-FV outperforms GMM-FV, since the probability distribution of binary local features is better described using mixtures of Bernoulli rather than mixtures of Gaussian. The results of our experiments also show that the use of BMM-FV is still effective even if compared with the direct matching strategy. In fact, the retrieval performance of BMM-FV on Oxford5k is just slightly worse than the traditional direct matching of local features, while on INRIA Holidays the BMM-FV even outperforms the direct matching result.

In conclusion, it is important to point out that there are several applications where binary features need to be used to improve efficiency, at the cost of some effectiveness reduction [126]. We showed that in this case, the use of the encodings techniques is a valid alternative to the direct matching.

3.2.3.3 Combination of CNNs and Aggregations of Binary Local Feature

In Section 3.1 we experimentally have shown that the information provided by the FV built upon SIFT helps to further improve the retrieval performance of the CNN features and thus a combination of FV and CNN features can be profitably used for instance retrieval tasks, as also done in [63]. However, the benefits of such combinations are clouded by the cost of extracting SIFTs that can be considered too high with respect to the cost of computing the CNN features, especially for systems that need to process images in real time. For example, according to [126] and as showed in the table 3.12, the SIFTs extraction (about $2,000$ features per image), the PCA-reduction to $D = 64$ dimensions, and the FV

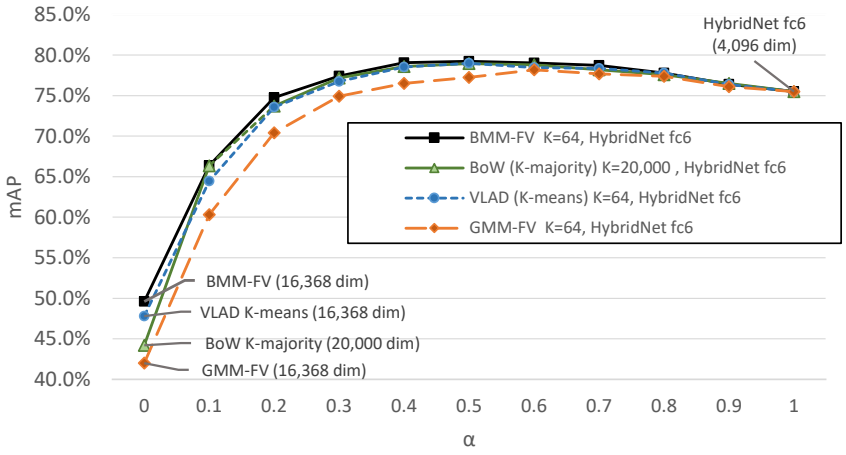


Figure 3.8: *Retrieval performance on the INRIA Holidays dataset of the combination of HybridNet* fc6 *and various aggregations of ORB binary feature (BMM-FV, VLAD, BoW, and GMM-FV). Only the full-sized descriptors are considered (*i.e.*, no PCA) and for each aggregation technique we selected the corresponding best setting (e.g learning method) according with results reported in Table 3.13.* α *is the parameter used in the combination:* $\alpha = 0$ *corresponds to use only the aggregated descriptor, while* $\alpha = 1$ *correspond to use only the HybridNet feature.*

aggregation with $K = 256$ compressively requires more than 1.3 seconds per image, while the CNN feature extraction is 4 times faster (*i.e.*, about 300 ms per image). On the other hand, extracting ORB binary features (about $2,000$ features per image, each of dimension $D = 256$) and aggregating them using a BMM-FV with $K = 64$ requires less than 190 ms that is in line with the cost of CNN extraction on CPU (300 ms).

Since the extraction of binary local features is up to two orders faster than SIFT, in the following we investigate the combination of CNN features with aggregations of binary local features, including BMM-FV. The combination was computed as described in the Section 3.1.1. Please note that in this case both the FV and the CNN features are ℓ_2 normalized so the distance function between the CNN descriptors has the same range value of the distance function between the BMM-FV descriptors. In our tests, the cost of integrating the already extracted BMM-FV and the CNN features was negligible in the search phase, using a sequential scan to search a dataset, also thanks to the fact that both BMM-FV and CNN features are compared using the not too costly Euclidean distance. For reference, combinations between CNN features and other image descriptors, such as GMM-FV, VLAD, and BoW are considered as well by using the convex combination of the respective distances. Whenever the range of the two used distances was not the same, the distances were rescaled before the convex combination.

For this set of experiments, we considered the INRIA Holidays dataset and we used the the output of the first fully-connected layer (*fc6*) of the HybridNet [276] model as CNN feature. In fact, in [63] several experimental results have shown that on the INRIA Holidays the *HybridNet fc6* achieves better mAP than other outputs (*i.e. pool5, fc6, fc7, and fc8*) of several pre-trained CNN models (OxfordNet, AlexNet, PlacesNet, and HybridNet).

Figure 3.8 shows the mAP obtained by combining HybridNet *fc6* with different aggregations of ORB binary features, namely the BMM-FV, the GMM-FV, the VLAD, and the BoW. Interestingly, with the exception of the GMM-FV, the retrieval performance obtained after the combination is very similar for the various aggregation techniques. This, on the one hand, confirms that the GMM-FV is not the best choice for encoding *binary* features; on the other hand, since each aggregation technique computes

Table 3.13: *Retrieval performance of various combinations of BMM-FV and HybridNet CNN feature. The BMM-FV representations were computed for three different binary local features (ORB, LATCH, and A-KAZE) using $K = 64$ mixtures of Bernoulli. The CNN feature was computed as the output the HybriNet fc6 layer after applying the ReLU activation function.* Dims *is the number of components of each vector representation. α is the parameter used in the combination of FV and CNN: $\alpha = 0$ corresponds to use only FV, while $\alpha = 1$ correspond to use only the HybridNet feature. Bold numbers denote maxima in the respective column.*

Method	**Dims**			$\boldsymbol{\alpha}$	**mAP** (%)		
	ORB	LATCH	A-KAZE		ORB	LATCH	A-KAZE
BMM-FV (K=64)	16,384	16,384	32,768	0	49.6	46.3	43.7
Combination of *BMM-FV (K=64)* and *HybridNet fc6*	20,480	20,480	36,864	0.1	66.4	64.7	59.2
				0.2	74.8	73.8	68.7
				0.3	77.4	76.8	74.3
				0.4	79.1	77.5	77.3
				0.5	**79.2**	78.3	78.0
				0.6	79.0	**78.5**	**79.2**
				0.7	78.7	77.7	78.7
				0.8	77.8	76.7	77.5
				0.9	76.4	76.3	76.2
HybridNet fc6		4,096		1		75.5	

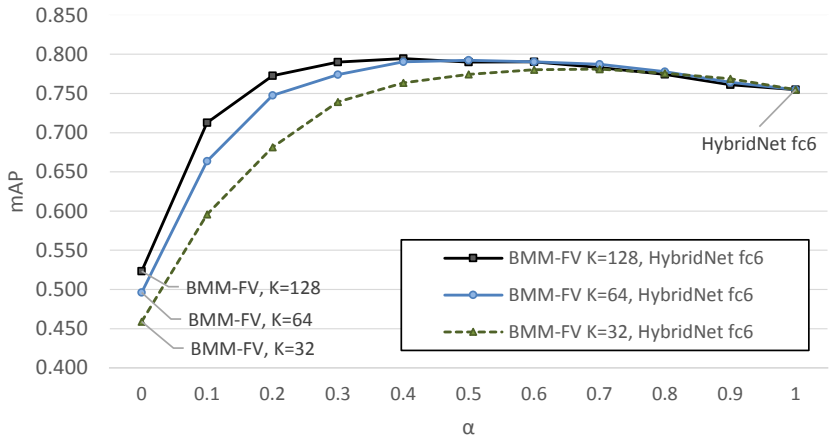


Figure 3.9: *Retrieval performance of the combinations of BMM-FV and HybridNet* fc6 *for various number K of Bernoulli mixtures. The BMM-FVs were computed using ORB binary features. α is the parameter used in the combination: $\alpha = 0$ corresponds to use only FV, while $\alpha = 1$ correspond to use only the HybridNet feature.*

Table 3.14: *Comparison of the results obtained combining HybridNet fc6 feature with the full-sized and the PCA-reduced versions of the BMM-FV. The BMM-FV was computed on ORB binary feature using $K = 64$ mixtures of Bernoulli.* Dim *is the number of components of each vector representation. α is the parameter used in the combination of FV and CNN. Bold numbers denote maxima in the respective column.*

Method	**Dims**		α	**mAP** (%)	
	FV full dim	FV PCA-reduced		FV full dim	FV PCA-reduced
BMM-FV (K=64)	16,384	4,096	0	49.6	52.6
Combination of *BMM-FV (K=64)* and *HybridNet fc6*	20,480	8,192	0.1	66.4	66.3
			0.2	74.8	73.9
			0.3	77.4	77.3
			0.4	79.1	**78.5**
			0.5	**79.2**	78.4
			0.6	79.0	78.5
			0.7	78.7	78.1
			0.8	77.8	77.7
			0.9	76.4	76.4
HybridNet fc6	4,096		1	75.5	

statistical summaries of the *same set* of the local descriptors, our results suggest that the additional information provided by the various aggregated descriptors helps almost equally to improve the retrieval performance of the CNN feature.

In the following we further investigate combinations of CNNs and the BMM-FV that, even for a shot, reaches the best performance for all the tested parameter α. In Table 3.13 we report the mAP obtained combining the HybridNet fc6 feature with the BMM-FV computed for three different kind of binary local features: ORB, LATCH and A-KAZE. It is worth noting that all the three BMM-FVs give a similar improvement when combined with the HybridNet fc6, although they have rather different mAP results (see the first row of Table 3.13) which are substantially lower than that of CNN (last row of Table 3.13). The intuition is that the additional information provided by using a specific BMM-FV rather than using the CNN feature alone, do not depend very much on the used binary feature.

For each tested BMM-FV seems that exist an optimal α to be used in the convex combination. When ORB binary features were used, the optimal α was obtained around 0.5, which correspond to give the same importance to both FV and CNN feature. For the less effective BMM-FVs built upon LATCH and A-KAZE, the optimal α was 0.6, which means that the CNN feature is used with slightly more importance than BMM-FV during the convex combination.

The use of ORB or A-KAZE led to obtain the best performance that was 79.2% of mAP. This results in a relative improvement of 4.9% respect to the single use of the CNN feature, that in our case was 75.5%. So we obtain the same relative improvement of [63] but using a less expensive FV representation. Indeed, in [63] the fusion of HybridNet *fc6* and a FV computed on 64-dimensional PCA-reduced SIFTs, using $K = 256$ mixtures of Gaussian, have led to a relative improvement of 4.9% respect to the use of the CNN feature alone.

Since as observed in [17, 166] the ORB extraction is faster than LATCH and A-KAZE, in the following we focus only on ORB binary feature. In figure 3.9 we show the results obtained by combining HybridNet *fc6* with the BMM-FVs obtained using $K = 32, 64, 128$. We observed that the performance of the CNN feature is improved also when it is combined with the less effective BMM-FV built using $K = 32$ Bernoulli. The BMM-FV with $K = 128$ achieve the best effectiveness (mAP of 79.5%) for $\alpha = 0.4$. However, the improvement obtained using $K = 128$ respect to that of $K = 64$ does not worth the extra cost of using a bigger value of K.

The BMM-FV with $K = 64$ is still high dimensional, so to reduce the cost of storing and comparing FV, we also evaluated the combination after the PCA dimensionality reduction. As already observed,

Table 3.15: *Relative mAP improvement obtained after combining FV with HybridNet* fc6*. Each relative improvements was computed respect to the use of the CNN feature alone, that is:* $(mAP_{\textit{after combination}} - mAP_{\textit{HybridNet fc6}})\,/mAP_{\textit{HybridNet fc6}}$*. The relative improvement obtained using the FV encoding of the SIFTPCA64 features was computed according to the results reported in [63].*

FV method method	Local Feature	K	Dims	Relative improvement
BMM-FV	ORB	128	32,768	5.2
BMM-FV	ORB	64	16,384	4.9
BMM-FV	A-KAZE	64	32,768	4.9
BMM-FV	LATCH	64	16,384	4.0
BMM-FV+ PCA	ORB	64	4,096	3.9
BMM-FV	ORB	32	8,192	3.5
FV [63]	SIFTPCA64	256	32,768	4.9

limited dimensionality reduction tends to improve the accuracy of the single FV representation. However, as shown in Table 3.14 and Table 3.15, when the PCA-reduced version of the BMM-FV was combined with HybriNet *fc6*, the overall relative improvement in mAP was 3.9%, which is less than that obtained using the full-sized BMM-FV. This result is not surprising given that after the dimensionality reduction we may have a loss of the additional information provided by the FV representation during the combination with the CNN feature.

Finally, in Table 3.15 we summarize the relative improvement achieved by combining BMM-FV and HybriNet *fc6*, and we compare the obtained results with the relative improvement achieved in [63], where the more expensive FV built upon SIFTs was used. We observed that BMM-FV led to achieve similar or even better relative improvements with an evident advantage from the computational point of view, thanks to the use of binary local features.

3.2.3.4 Large-Scale Experiments

In order to evaluate the behaviour of feature combinations on a large scale, we have used a set of up to one million images. More precisely, as in [139], we merged the INRIA Holidays dataset with a public large-scale dataset (MIRFlickr-1M [135]) used as distraction set; the mAP was measured using the Holidays ground-truth.

Table 3.16 shows the results obtained using both the BMM-FV alone and the combinations with the *HybridNet fc6* CNN feature. Given the results reported in the previous sub-sections we focus on the BMM-FV encoding of ORB binary features. All the feature combinations show an improvement with respect to the single use of the CNN feature (mAP of 59.1%) or BMM-FV (mAP of $31.0\%/34.9\%$ respectively using the full length/PCA-reduced descriptor). This reflects the very good behaviour of feature combinations also in the large-scale case.

The mAP reaches a maximum using α between 0.4 and 0.5, that is giving (quite) the same weight to BMM-FV and CNN feature during the combination. The results obtained using the full-length BMM-FV and the PCA-reduced version are similar. Surprisingly, the latter performs slightly better and achieved a maximum of 67.2% of mAP that correspond to 13.7% of relative mAP improvement respect to use the CNN feature alone. It is worth noting that the relative mAP improvement obtained in the large-scale setting is much greater than that obtained without the distraction set. This suggests that the information provided by the local features during the combination helps in discerning the visual content of images particularly in presence of distractor or noisy images.

Since the computational time of extracting binary features is much faster than others, the computational gain of combining CNN features with BMM-FV encodings of ORB over traditional FV encodings of SIFT is especially notable in the large-scale scenario. For example, the process for extracting SIFTs from the INRIA Holidays+ MIRFlickr dataset (1,001,491 images) would have required more than 13 days (about 1,200 ms per image) while ORB extraction took less than 8 hours (about 26 ms per image).

Table 3.16: *Comparison of the results obtained combining HybridNet fc6 feature and BMM-FV on the INRIA Holidays dataset with the distractor dataset MIRFlickr-1M. The results related to the INRIA Holidays alone are reported from Table 3.14 for reference. The BMM-FV was computed on ORB binary feature using $K = 64$ mixtures of Bernoulli; both full-sized and the PCA-reduced features are considered.* Dim *is the number of components of each vector representation. α is the parameter used in the combination of FV and CNN. Bold numbers denote maxima in the respective column. The last row reports the maximum relative mAP improvement obtained after combining HybridNet* fc6 *with FVs.*

Method	**Dims**		α	**mAP** (%)			
	FV full dim	FV PCA-reduced		FV full dim		FV PCA-reduced	
				Holidays	Holidays+ MIRFlickr	Holidays	Holidays+ MIRFlickr
BMM-FV (K=64)	16,384	4,096	0	49.6	31.0	52.6	34.9
Combination of *BMM-FV (K=64)* and *HybridNet fc6*	20,480	8,192	0.1	66.4	47.0	66.3	50.7
			0.2	74.8	59.3	73.9	61.9
			0.3	77.4	64.0	77.3	65.6
			0.4	79.1	**67.1**	**78.5**	**67.2**
			0.5	**79.2**	66.5	78.4	66.9
			0.6	79.0	65.7	78.5	65.7
			0.7	78.7	64.4	78.1	64.4
			0.8	77.8	62.5	77.7	62.8
			0.9	76.4	60.7	76.4	60.8
HybridNet fc6	4,096		1	75.5	59.1	75.5	59.1
Maximum relative mAP improvement $\rightarrow$				4.9%	13.4%	4.0%	13.7%

3.2.4 Summary

Motivated by recent results obtained on one hand with the use of aggregation methods applied to local descriptors, and on the other with the definition of binary local features, in this section we performed an extensive comparison of techniques that combine the two approaches by using aggregation methods on binary local features. The use of aggregation methods on binary local features meets the need to increase the efficiency and reduce the computing resources for image matching, at the expense of some degradation in the accuracy of retrieval algorithms. Mixing the two approaches lead to execute image retrieval on a large scale and reduce the cost for feature extraction and representation. Therefore we expect that the results of our study are useful for people that for efficiency issues work with binary local descriptors.

Moreover, we investigated how aggregations of binary local features work in conjunction with the CNN features in order to improve the latter retrieval performance. We showed that our BMM-FV built upon ORB binary features can be profitable use to this scope, even if a relatively small number of Bernoulli is used. In fact, the relative improvement in the retrieval performance obtained combining CNN features with the BMM-FV is similar to that previously obtained in the literature using a combination of the CNN features with the more expensive FV built on SIFT. Experimental evaluation on large-scale confirms the effectiveness and scalability of our proposal.

It is also worth mentioning that our BMM-FV approach is very general and could be applied to encode any set of binary vectors, also not related to the image search scenario.

CHAPTER 4

Features Processing for Efficient Indexing

In order to perform image retrieval on large scale database, the descriptors extracted from images need to be indexed. Many mechanisms are suitable for this scope, notably including metric indexes [36, 65, 198, 200]. The high extensibility of the metric search framework allows handling features of any nature, supposing that there exists a metric on the feature domain. However, it is sometimes the case that structural or semantic peculiarities of the image features can be further exploited to "specialize" metric approaches in order to improve the performance on a specific task or even to use off-the-shelf indexing/searching mechanisms with a little implementation effort. For example, the Surrogate Text Representation (STR) [109] technique (introduced in Section 2.4.9.3) transforms a global descriptor into a textual form in order to use conventional text search engine. Following this research direction, the present Chapter proposes two promising approaches for processing some state-of-the-art image features, like VLAD and CNN features.

In Section 4.1 we extend STR approach to address a specific class of features that have a block vector representation, such as VLADs and FVs. In fact, VLAD and FV descriptors, are obtained by concatenating sub-vectors, each capturing a different statistic of the image local descriptors. The proposed *Blockwise Surrogate Text Representation (BSTR)* transforms each sub-vector into a textual form by using a local codebook for each block instead of using a global codebook for the full vector. This is equivalent to represent the individual sub-vectors with the STR in order to provide a finer representation of the full vector. We tested this technique on VLAD descriptors extracted from a benchmark. Our experiments show that the BSTR outperforms the baseline STR achieving performance near to the one obtained using exact search on the original VLAD vectors. Moreover BSTR encodings of VLADs enabling us to get rid of the reordering phase, which was needed by STR to achieve satisfactory performance.

In Section 4.2 we focus on the emerging deep CNN features, which have been successfully used in generic recognition and visual search tasks. CNN features typically are of high dimensionality. This represents a major obstacle to their use on large scale, due to the well-known curse of dimensionality. A promising approach to tackle this problem is using approximate access methods, as for instance permutation-based approaches [36, 64, 96, 200] (see Section 2.4.9.2). PBI methods represent metric objects as sequences (permutations) of pivots, chosen from a predefined set of objects. Each permutation is traditionally generated by sorting the entire set of reference objects according to their distances from the object to be represented. The optimal number of reference objects depends on characteristics (*e.g.* size and intrinsic dimensionality) of the dataset to be indexed as well as on the used indexing approach. For

example, in the MI-File [36] it can amount to tens of thousands. In such case, both indexing and searching times are highly affected by the cost of generating permutations. To address this issue we propose an approach to generate permutations for deep features at a very low computational cost. The advantage of our method, called *Deep Permutations*, is twofold: 1) it does not require the distance calculations between the reference objects and the objects to be represented, 2) the obtained "deep" permutations results more effective than those obtained using pivot selection criteria specifically developed for permutation-based methods.

Main results presented in this Chapter were published in [20, 29]. Table 4.1 summarizes the used notation.

4.1 BSTR: Blockwise Surrogate Text Representation

The STR is a useful solution to perform similarity search by exploiting text search engine. It encodes a descriptor with a text document by using a permutation-based representation as intermediate step. The text document is then indexed using a text search engine. The distance between the text documents is a coarse approximation of the actual distance between objects. Therefore, as recommended for many other approximate methods, the result set obtained using STR need to be reordered according to the actual distance to achieve high accuracy.

In this section, we extend the STR approach to deal with descriptors that have a block vector representation, such as VLADs. The underline idea is to use a permutation-representation for each block of the vector in order to have a finer representation that allows us to get rid of the reordering phase. We first prove the equivalence of the baseline STR with the permutation representation (Section 4.1.1). Then, we present our BSTR technique and shows its equivalence with a blockwise-permutation representation (Section 4.1.2). Later, we discuss how use the BSTR encoding to index VLAD vectors (Section 4.1.3). Finally we show experimental results (Section 4.1.4).

4.1.1 STR and Permutation-based Approach

We recall that given metric space $(\mathcal{D}, d)$ and a set of pivots $\{p_1, \ldots, p_n\}$ we can represents any object $o \in \mathcal{D}$ as a vector Π_o^{-1} of rank positions of the pivots, ordered on the basis of their distance from o. Typically a top-l list is considered, which contains rankings for only a subsets of pivots (the closest ones). In such case, a location parameter l is considered and an inverted truncated permutation $\Pi_{o,l}^{-1}$ is used to represents data objects. Several top-l distances [97] can be used for the comparison. Here we focus on that based on Spearman rho distance for two main reasons: 1) it showed good performance in several metric spaces [64], 2) (as explained later) the Spearman rho is tied to the way standard search engines process the similarity between documents and queries. In particular, the Spearman rho with location parameter l compares two top-l ranked lists by assigning a rank $l+1$ for all items of the list that have rank greater than l. Similarly, given two objects o and q, and the associated top-l lists, we consider the approximate distance $\tilde{d}(o, q)$ as follows:

$$\tilde{d}(o, q) = \ell_2 \left(\Pi_{o,l_o}^{-1}, \Pi_{q,l_q}^{-1} \right), \tag{4.1}$$

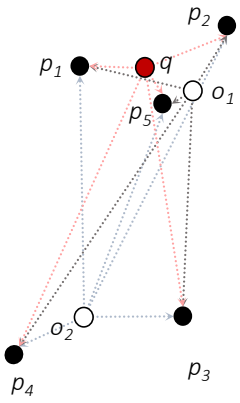
where l_q is used for queries and l_o for indexing, with $l_q < l_o$. The reason for using two different location parameter relies on the fact that the performance of the inverted files (used to index the texts) is optimal when the size of the queries are much smaller than the size of documents. Note that the only differences between $\tilde{d}$ and the traditional Spearman's rho with location parameter is the domain of definition (data objects instead of permutations) and the use of two different location parameters. Actually, $\tilde{d}$ is not a metric on $\mathcal{D}$ but it is a dissimilarity function.

The STR representation $\mathrm{t}_{x,l}$ of an object $x \in \mathcal{D}$ is obtained as space-separated concatenation of some alphanumeric keywords. The latter are selected from a dictionary $\{\tau_1, \ldots, \tau_n\}$, where each keyword is uniquely associated with a pivot. Formally,

$$\mathrm{t}_{x,l} = \bigcup_{i=1}^{n} \bigcup_{j=1}^{l+1-\Pi_{x,l}^{-1}(i)} \tau_i, \tag{4.2}$$

Table 4.1: *Notation used throughout this Chapter*

Symbol	Definition
$(\mathcal{D}, d)$	metric space
$\mathcal{S}$	finite search space, $\mathcal{S} \subseteq \mathcal{D}$
N	number of objects in $\mathcal{S}$
$\{p_1, \ldots, p_n\}$	set of pivots, $p_i \in \mathcal{D}$
n	number of pivots
o	data objects, $o \in \mathcal{S}$
q	query, $q \in \mathcal{D}$
l	generic location parameter (permutation prefix)
l_o	location parameter used for the data objects
l_q	location parameter used for the query object
Π_x	pivot permutation: $\Pi_x(i)$ is the *pivot identifier* at position i in the ranked list of the nearest pivots to the object $x \in \mathcal{D}$
Π_x^{-1}	inverted permutation: $\Pi_x^{-1}(i)$ is the *position* of the pivot p_i in the ranked list of the nearest pivots to the object $x \in \mathcal{D}$
$\Pi_{x,l}^{-1}$	inverted truncated permutation (values of Π_x^{-1} bigger than l are replaced with the constant value $l+1$)
$\{\tau_1, \ldots, \tau_n\}$	codebook, τ_i is an alphanumeric keyword associated with the pivot p_i
t_x	text encoding (document) associated to the object x
$\mathrm{t}_{x,l}$	text encoding (document) associated to the object x if using a location parameter l
$\mathbf{p}_x$	vector of occurrences of the keywords τ_i in the document t_x
$\mathbf{p}_{x,l}$	vector of occurrences of the keywords τ_i in the document $\mathrm{t}_{x,l}$
$sim_{\cos}(o, q)$	approximation of the actual similarity between the object o and q by means of the cosine similarity between the documents t_{o,l_o} and t_{q,l_q}
$\tilde{d}(o, q)$	approximation of the actual distance $d(o, q)$ by means of the Spearman's rho distance between the corresponding permutation-representations
ℓ_2	Euclidean distance
$\mathbf{e} = [1, \ldots, 1]$	constant vector
$\mathbf{V} = [v_1, \ldots, v_K]$	blockwise object, $v_i \in \mathcal{D}$
$\mathbf{Q} = [q_1, \ldots, q_K]$	blockwise query,$q_i \in \mathcal{D}$
$\mathbf{B_X} = [\Pi_{x_1,l_x}^{-1}, \ldots, \Pi_{x_K,l_x}^{-1}]$	blockwise inverted permutation associated to a blockwise vector $\mathbf{X} = [x_1, \ldots, x_K]$, $x_1 \in \mathcal{D}$
$\hat{d}(\mathbf{V}, \mathbf{Q}) = \ell_2(\mathbf{B_V}, \mathbf{B_Q})$	approximation of the actual distance $d(\mathbf{V}, \mathbf{Q})$ by means of the Euclidean distance between the blockwise inverted permutations
$\tilde{\Pi}_{\mathbf{v}} = [\tilde{\Pi}_{\mathbf{v}}(1), \ldots, \tilde{\Pi}_{\mathbf{v}}(n)]$	deep permutation associated to a vector $\mathbf{v} \in \mathbb{R}^n$



Codebook: τ_1 = "A", τ_2 = "B", τ_3 = "C", τ_4 = "D", τ_5 = "E"		
$\Pi_{o_1} = [5,2,1,3,4]$	$\Pi_{o_2} = [4,3,5,1,2]$	$\Pi_q = [5,1,2,3,4]$
$\Pi^{-1}_{o_1} = [3,2,4,5,1]$ (p_1,p_2,p_3,p_4,p_5)	$\Pi^{-1}_{o_2} = [4,5,2,1,3]$ (p_1,p_2,p_3,p_4,p_5)	$\Pi^{-1}_{q} = [2,3,4,5,1]$ (p_1,p_2,p_3,p_4,p_5)
$l_o = 3$		$l_q = 2$
$\Pi^{-1}_{o_1,l_o} = [3,2,4,4,1]$	$\Pi^{-1}_{o_2,l_o} = [4,4,2,1,3]$	$\Pi^{-1}_{q,l_q} = [2,3,3,3,1]$
$\mathbf{p}_{o_1,l_o} = [1,2,0,0,3]$ ($\tau_1,\tau_2,\tau_3,\tau_4,\tau_5$)	$\mathbf{p}_{o_2,l_o} = [0,0,2,3,1]$ ($\tau_1,\tau_2,\tau_3,\tau_4,\tau_5$)	$\mathbf{p}_{q,l_q} = [1,0,0,0,2]$ ($\tau_1,\tau_2,\tau_3,\tau_4,\tau_5$)
t_{o_1,l_o} = "E E E B B A"	t_{o_2,l_o} = "D D D C C E"	t_{q,l_q} = "E E A"

Figure 4.1: *Example of perspective-based space transformation. Black points are reference objects; white points are data objects; the red point is a query. From the top to the bottom line: chosen codebook, permutation representations, inverted permutations, location parameter values, inverted truncated permutations, term frequencies vectors, textual encodings (STRs).*

where we denote the space-separated concatenation of keywords with the union operator $\cup$. Therefore, the vector of the *term frequencies* associated to the document $t_{x,l}$ is

$$\mathbf{p}_{x,l} = (l+1)\mathbf{e} - \Pi^{-1}_{o,l} \tag{4.3}$$

where $\mathbf{e} = [1, \ldots, 1]$ is the constant m-dimensional vector.

Figure 4.1 exemplifies the transformation process. On the left, it sketches a number of pivots (black points), data objects (white points), and a query object (red point). On the right, it shows the encodings of the objects both as permutations and text documents. In the case showed in Figure, the STRs of o_1, o_2, and q are

$$\begin{aligned} t_{o_1,l_o} &= \text{“E E E B B A”} \\ t_{o_2,l_o} &= \text{“D D D C C E ”} \\ t_{q,l_q} &= \text{“E E A”}. \end{aligned}$$

It can be notice that the strings corresponding to o_1 and q are intuitively more similar to those corresponding to o_2 end q. This reflects the behaviour of the distance $\tilde{d}$, for which o_1 is closer to q than o_2.

Actually, the built-in cosine similarity measure of standard text-based search engines (based on the vector space model) exactly corresponds to $\tilde{d}$. In facts, a text based search engine will generate a vector representation of STRs that contains the number of occurrences of words in texts. In the case of the simple term-frequency weighting scheme, the vector is $\mathbf{p}_{x,l}$. This means that, if the keyword τ_i appears k times in the text, the i-th element of the vector $\mathbf{p}$ will assume the value k, and whenever τ_i does not appear, it will be 0. It is worth noting that since $\mathbf{p}_{x,l}$ is a permutation of the n-dimensional vector $[1, 2, \ldots, l, 0, \ldots, 0]$, its euclidean norm is constant and equals to $\sqrt{l(l+1)(2l+1)/6}$. Thus, for fixed l_o and l_q, the norms of vectors $\mathbf{p}_{o,l_o}$ and $\mathbf{p}_{q,l_q}$ are constants too, which can be neglected during the cosine evaluation (they do not affect the final ranking of the search result). The cosine similarity, typically adopted to determine the similarity of the query vector and a vector in the database of the search engine, in this specific case will be proportional to the dot product of the term frequencies vectors:

$$sim_{\cos}(o,q) = \frac{\mathbf{p}_{o,l_o} \cdot \mathbf{p}_{q,l_q}}{\|\mathbf{p}_{o,l_o}\| \ \|\mathbf{p}_{q,l_q}\|} \propto \mathbf{p}_{o,l_o} \cdot \mathbf{p}_{q,l_q}. \tag{4.4}$$

What we are going to show is that $sim_{\cos}$ can be used as a function for evaluating a similarity of two objects in place of the dissimilarity function $\tilde{d}$ and that the first one is an order reversing monotonic

transformation of the second one (they are equivalent for practical aspects). This means that if we use $\tilde{d}(o,q)$ and we take the first k nearest objects from a dataset $\mathcal{S} \subset \mathcal{D}$ (i.e, from the shortest distance to the highest) we obtain exactly the same objects in the same order of the result list obtained using $sim_{\cos}(o,q)$ and taking the first k similar objects (*i.e.*, from the greater values to the smaller ones).
By substituting Eq. (4.3) into Eq. (4.4), we obtain:

$$sim_{\cos}(o,q) \propto ((l_o+1)\mathbf{e} - \Pi^{-1}_{o,l_o}) \cdot ((l_q+1)\mathbf{e} - \Pi^{-1}_{q,l_q}) = \tag{4.5}$$

$$= (l_o+1)(l_q+1)\mathbf{e}\cdot\mathbf{e} - (l_o+1)\mathbf{e}\cdot\Pi^{-1}_{q,l_q} - (l_q+1)\mathbf{e}\cdot\Pi^{-1}_{o,l_o} + \Pi^{-1}_{o,l_o}\cdot\Pi^{-1}_{q,l_q}. \tag{4.6}$$

Since the generic permutation $\Pi^{-1}_{x,l}$ includes all integers numbers from 1 to l and the remaining assumes $l+1$, the scalar product $\Pi^{-1}_{x,l}\cdot\mathbf{e}$ is a constant for a fixed value of l. [1] Therefore, we can substitute the first three member in Eq. (4.6) with a constant $c(n,l_o,l_q)$, which depends only on n, l_o, and l_q as follows:

$$sim_{\cos}(o,q) \propto c(n,l_o,l_q) + \Pi^{-1}_{o,l_o}\cdot\Pi^{-1}_{q,l_q}. \tag{4.7}$$

By notice that

$$\tilde{d}(o,q)^2 = \sum_{i=1}^{n}\left(\Pi^{-1}_{o,l_o}(i) - \Pi^{-1}_{q,l_q}(i)\right)^2 = ||\Pi^{-1}_{o,l_o}||_2^2 + ||\Pi^{-1}_{q,l_q}||_2^2 - 2\,\Pi^{-1}_{o,l_o}\cdot\Pi^{-1}_{q,l_q} \tag{4.8}$$

we finally obtain

$$sim_{\cos}(o,q) \propto c(n,l_o,l_q) + \frac{1}{2}||\Pi^{-1}_{o,l_o}||_2^2 + \frac{1}{2}||\Pi^{-1}_{q,l_q}||_2^2 - \frac{1}{2}\tilde{d}(o,q)^2. \tag{4.9}$$

Since also $||\Pi^{-1}_{o,l_o}||_2$ and $||\Pi^{-1}_{q,l_q}||_2$ depend only on the constants n, l_o, and l_q, the Eq. (4.9) proves that $sim_{\cos}(o,q)$ is a monotonic transformation of $\tilde{d}(o,q)^2$ in the form $sim_{cos} = \alpha - \beta\tilde{d}^2$, with $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}^+$.

In summary, we can consider an approximate distance function $\tilde{d}$ of a given a metric d by exploiting the permutation-based representation, as done by many PBI approaches. We are able to transform such distance into a similarity measure between text representations (the STRs), such that the two measures are equivalent from the point of view of the ranking results.

Given the proved equivalence between text and permutation representations, the main motivation for using the STR representation rather than classic PBI approaches is, on one hand, to exploit off-the-shelf software library with minimal implementation efforts and, on the other hand, to benefit from advanced functionality and optimization procedure available for text retrieval.

4.1.2 BSTR and Blockwise Permutation-Based Approach

The idea described so far uses a textual/permutation representation of an object as a whole and can be applied in every metric space. We now focus on a simple consequence: when dealing with objects that are compound of other metric objects, we can apply the permutation representation to each compound, independently. We call this approach *Blockwise Permutation-based representation*.

We defined and tested this approach in the case of blockwise vectors, such as the VLAD descriptors that are the result of the concatenation of sub-vectors. The proposed BSTR is less general than STR, but when applicable it provides a finer representation of the data in terms of text/permutation. The main objective here is improving the quality of the search result in order to avoid the reordering phase.

Let's assume that each data vector $\mathbf{V}$ is the concatenation of K object, *i.e.* $\mathbf{V} = [v_1,\ldots,v_K]$, where $v_i \in \mathcal{D}$. For each block v_j, $1 \le j \le K$, we consider a set of pivots $P_j = \{p_{1j},\ldots,p_{nj}\}$ to build the corresponding permutation representation. In this way we can represent each vector $\mathbf{V}$ as a concatenation of permutation vectors: $\mathbf{B_V} = [\Pi^{-1}_{v_1,l_o},\ldots,\Pi^{-1}_{v_K,l_o}]$.

[1] $\Pi^{-1}_{x,l}\cdot\mathbf{e} = \sum_{i=1}^{l} i + \sum_{l+1}^{n}(l+1) = l(l+1)/2 + (l+1)(n-l)$

So, given two blockwise vector $\mathbf{V} = [v_1, \ldots, v_K]$ and $\mathbf{Q} = [q_1, \ldots, q_K]$, we define the dissimilarity function

$$\hat{d}(\mathbf{V}, \mathbf{Q})^2 = \sum_{j=1}^{K} \tilde{d}(v_j, q_j)^2 = \sum_{j=1}^{K} \ell_2(\Pi^{-1}_{v_j,l_o}, \Pi^{-1}_{q_j,l_q})^2 = \ell_2\left(\mathbf{B_V}, \mathbf{B_Q}\right)^2 \tag{4.10}$$

This "generalization" of the Spearman rho can be efficiently computed as the Euclidean distance of the concatenated permutation vectors.

In order to generate a surrogate text representation that correctly matches the transformed blockwise vectors, we need to extend the reference dictionary to distinguish the key produced from each block. So for the j-th block we consider the codebook $\{\tau_{1j}, \ldots, \tau_{nj}\}$. In total we have a dictionary of $n \times K$ keywords. For example, we associate, say, the set of keys A_1, B_1,... to the first block, A_2, B_2,... to the second block, and so on.

The text associated to a data vector is the space-separated concatenation of the text associated to each block. Since we used a different codebook for each block, the term frequency vectors $\mathbf{p_V}$ and $\mathbf{p_V}$ equals $[\mathbf{p}_{v_1}, \ldots, \mathbf{p}_{v_K}]$ and $[\mathbf{p}_{q_1}, \ldots, \mathbf{p}_{q_K}]$ respectively, where

$$\begin{aligned} \mathbf{p}_{v_i,l_o} &= (l_o + 1)\mathbf{e} - \Pi^{-1}_{v_j,l_o} \\ \mathbf{p}_{q_i,l_q} &= (l_q + 1)\mathbf{e} - \Pi^{-1}_{q_j,l_q}. \end{aligned} \tag{4.11}$$

By a similar procedure shown above, it is immediate to prove that also in this case the relation

$$sim_{cos}(\mathbf{V}, \mathbf{Q}) = \alpha - \beta\, \hat{d}^2(\mathbf{V}, \mathbf{Q}) \tag{4.12}$$

holds for some constant $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}^+$ depending on l_o, l_q, K, n.

4.1.3 BSTR for VLAD Vectors

Here we consider the application of the BSTR encodings to VLAD descriptors. First of all, we observe that in this case each block a VLAD descriptor is a vector itself. In order to decrease the complexity of the approach and since VLAD sub-vectors $\mathbf{v}_i$ are homogeneous we use the same set of reference objects $\{p_1, \ldots, p_n\}$ to represent them as permutations. We select the pivots at random from the dataset of all VLAD sub-vectors.

We recall that given a codebook $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$, the VLAD sub-vector $\mathbf{v}_i$ encodes the accumulated difference between the visual codeword $\boldsymbol{\mu}_i$ and the associated local descriptors. So, one of the well-known problems of VLAD happens when no local descriptor is assigned to a visual codeword [204]. A simple approach to this problem is producing a sub-vector of all zeros ($\mathbf{v}_i = \mathbf{0}$) but this has the disadvantage to be ambiguous since it is identical to the case in which the mean of the local descriptors assigned to a codeword is equal to the codeword itself.

Moreover, as pointed out by [238], given two images and the corresponding VLAD vectors $\mathbf{V} = [\mathbf{v_1}, \ldots, \mathbf{v_K}]$ and $\mathbf{Q} = [\mathbf{q_1}, \ldots, \mathbf{q_K}]$, $\mathbf{v_i}, \mathbf{q_i} \in \mathbb{R}^D$, and assuming that $\mathbf{v}_i = \mathbf{0}$, the contribution of codeword $\boldsymbol{\mu}_i$ to the cosine similarity of $\mathbf{V}$ and $\mathbf{W}$ will be the same when either $\mathbf{w}_i = \mathbf{0}$ or $\mathbf{w}_i \neq \mathbf{0}$. Therefore, this under-estimates the importance of jointly zero components, which gives some limited yet important evidence on visual similarity [138]. In [138], this problem was treated by measuring the cosine between vectors $\mathbf{V}$ and $\mathbf{W}$ at different point from the origin. This technique, however, did not lead to significant improvement on our experiments. To tackle this problem, we simply get rid of the sub-vectors $\mathbf{v}_i = \mathbf{0}$ and omit to transform them into text. Mathematically, this means that we assume $\mathbf{p_0}_{,l_o} = \mathbf{0}$.

4.1.4 Experiments

This section reports the experimental evaluation of BSTR encoding of VLAD descriptors, comparing the effectiveness of the sequential scan and the state-of-the-art STR approach.

4.1.4.1 Experimental Setup

The experimentation was carried out on the INRIA Holidays [139] benchmark. To evaluate the approaches on a large scale, we merged the Holidays dataset with the MIRFlickr dataset [135], as done in Section 3.2.3.1 and many other articles [140, 141, 144].

In this set of experiments, we used the traditional VLAD, *i.e.* the VLAD aggregation of SIFT feature. We used the publicly available SIFTs extracted by Jegou et al. for both the Holidays and the MIRFlickr datasets[2]. The VLAD descriptors were computed using VIR [98]. The codebook $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$ was computed using *k-means*, with $K = 64$, over a subset of the 1 million MIRFlickr dataset. So at the end each VLAD has $8,192$ dimensions.

The observation that VLAD descriptors are relatively sparse and very structured suggests performing a principal component analysis to reduce the dimensionality of the descriptors. In this experiments, we decide not to use dimensionality reduction techniques because our space transformation approach is independent of the original dimensionality of the description.

The text representations were indexed and searched using Apache Lucene [3] that is an open-source information retrieval library. It delivers high-performance search features and it is suitable for nearly any application requiring full-text search abilities. All experiments were conducted on an Intel Core i7 CPU, 2.67 GHz with 12.0 GB of RAM a 2TB 7200 RPM HD for the Lucene index. We used Lucene v4.7 running on Java 6 64 bit.

The quality of the retrieved images was evaluated by means of mAP.

4.1.4.2 Results

In a first experimental analysis, we compared the retrieval performance of the BSTR approach to the STR approach that threats the VLAD vectors as whole-objects. The STR encoding of VLADs was previously studied by Amato et al. [23, 27]. As pointed in [27], since the retrieval performance of the baseline STR is low, we need to reorder the best results obtained from the text search engine using the actual distance between the VLAD descriptors. With this experiments, we want to show that the reordering phase is no longer necessary if using the blockwise approach and thus, the search results can be directly provided by the text-search engine Lucene.

In the following, we refer to baseline approach as STR, the baseline approach with reordering as rSTR, and the blockwise approach as BSTR. For the STR/rSTR approaches we used n =4,000 reference objects while for blockwise a total of n =20,000. For the rSTR approach, we reordered the first 1,000 objects of the results set. In all the cases, we set $l_o = 50$, which, we recall, is the number of closest pivots used during indexing.

Figure 4.2 shows the comparison in terms of mAP. The red horizontal line reports the performance (mAP of 55%) obtained by comparing the original VLAD descriptors by performing a sequential scan of the dataset (Exact search). Below this, the graph shows the mAP of our approach (BSTR) versus the STR approaches (with and without reordering) varying the location parameter l_q from 10 to 50 (number of closest pivots used for the query). An interesting by-product of the experiment is that the BSTR approach shows a local optimum for the mAP when the number of reference objects used for the query is 20. Figure 4.2 also reports result for an approach that we name "BSTR *tf-idf*". This exploits the knowledge of the *tf-idf* (*i.e.*, term frequency-inverse document frequency) statistic of the BSTR textual representation in order to further reduce the size of the query. So, instead of simply reducing the l_q of the query, *i.e.*, the top-l_q element nearest to the query, we can retain the elements that exhibit greater values of *tf-idf* and eliminate the others. Therefore, we take, for instance, the first 40 elements that have best *tf-idf*, the first 30 elements, and so on. Figure 4.2 shows the performance of this approach first considering a BSTR encoding with a location parameter equal to 50 and then selecting the l_q elements that have best *tf-idf*. It is interesting to note that with the *tf-idf* reduction we had not only an important improvement of the mAP for increasing reduction of the queries but also that this approach outperforms the performance of the exact search on the original VLAD dataset.

In order to ascertain the soundness of the proposed approach, we tested it on a larger and challenger dataset obtained by merging INRIA Holidays with the MIRFlickr dataset for a total of more than 1M

[2] `http://lear.inrialpes.fr/~jegou/data.php`
[3] `http://lucene.apache.org`

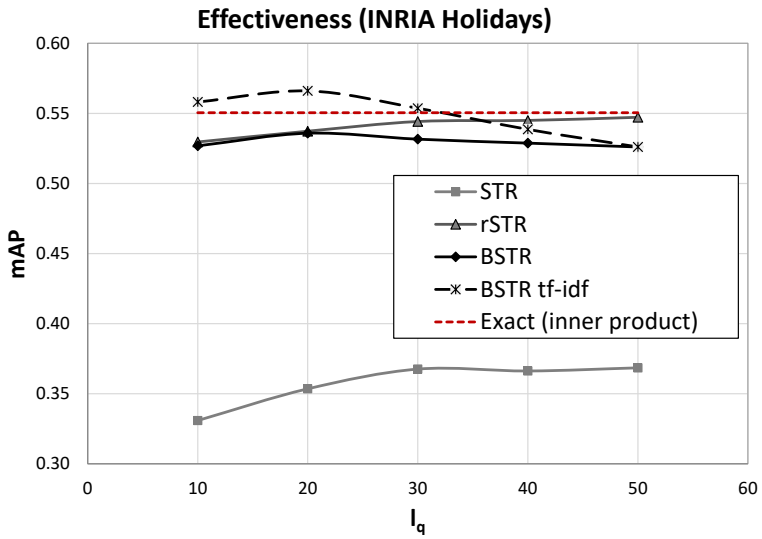


Figure 4.2: *Effectiveness (mAP) of the various approaches for the INRIA Holidays dataset, using $l_o = 50$ for STR, rSTR, BSTR, and BSTR tfidf (higher values mean better results).*

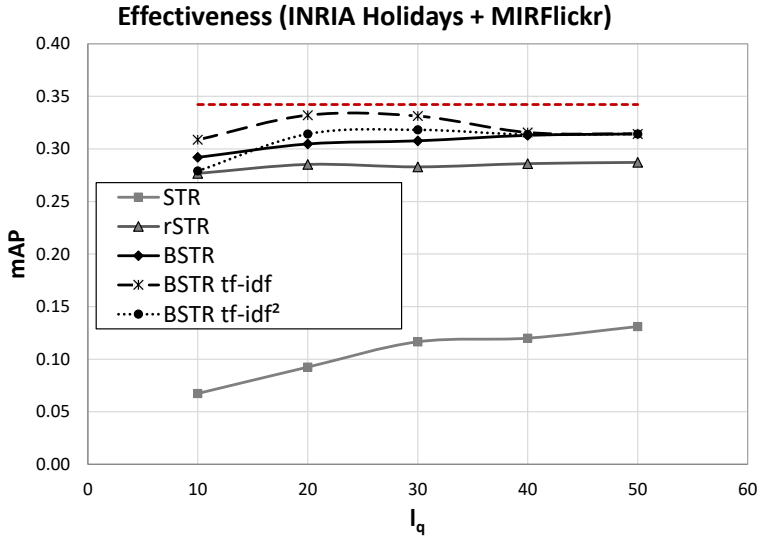


Figure 4.3: *Effectiveness (mAP) of the various approaches for the INRIA Holidays + MIRFlickr dataset, using $l_o = 50$ for STR, rSTR, BSTR, and BSTR* tf-idf. *While for BSTR* tf-idf^2*, we set $l_o = l_q$ (higher values mean better results).*

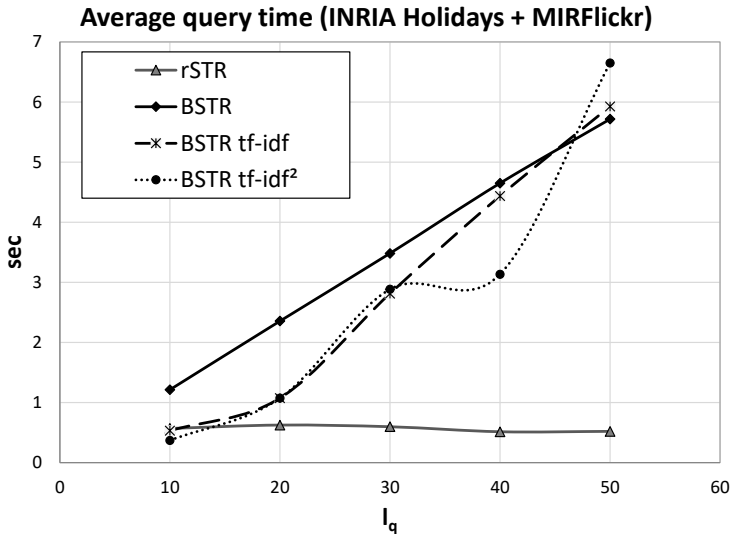


Figure 4.4: *Average time per query in seconds of the various approaches for the INRIA Holidays + MIRFlickr dataset, using $l_o = 50$ for rSTR, BSTR, and BSTR* tf-idf. *While for BSTR* tf-idf^2*, we set $l_o = l_q$ (higher values mean worse performance).*

images. The results are shown in Figure 4.3. We can see that BSTR *tf-idf* is still the winner in terms of mAP. However, in this case, all the techniques exhibit lower performance with respect the inner product on the original VLAD dataset. The latter test, performed as a sequential scan of the entire dataset, obtained a mAP of 0.34. The results presented in this figure also show the performance of the approach called "BSTR *tf-idf*^{2}", which consists in applying the reduction of the blockwise textual representation using *tf-idf* also for the indexed document (in addition to the queries), setting $l_o = l_q$ for all the experiments. The mAPs values, in this case, are slightly lower than BSTR *tf-idf*, however, as we are going to see in the next experiment there is a great advance in terms of space occupation.

In order to assess which approach is most promising, we have evaluated the efficiency in terms of space and time overhead. Figure 4.4 shows the average time for a query for the proposed approaches. Results for the rSTR approach includes also the time for reordering the result set. However, note that this average time was obtained using a solid state disk (SSD) disk in which the original VLAD vectors were available for the reordering. The SSD is necessary to guarantee fast random I/O, while if using a standard disk the seek time would affect the query time of more than one order of magnitude.

Figure 4.5 presents the index occupation expressed in GB. The rSTR approach occupies 16.8 GB on the disk, including the overhead for the storage of the VLAD vectors used for the reordering of the results. This last figure show how the *tf-idf* reduction of both queries and the indexed documents has a great impact on the space occupation: just for a reduction of the 20% of the documents (*i.e.*, from $l_o = 50$ to $l_o = 40$) we get a reduction of the 80% for the inverted file (see BSTR *tf-idf*^{2} results).

Considering all the alternatives seen so far in this applicative scenario, the BSTR *tf-idf*^{2} with $l_o = l_q = 20$ can be considered a trade-off between efficiency and effectiveness.

4.1.5 Summary

In this section, we proposed a blockwise permutation/text representation for compound metric objects. We tested the proposed BSTR approach for indexing VLAD vectors that, by definition, are obtained by concatenating some sub-vectors. The use of a textual encoding of the permutations allow us to use off-the-shelf text search engine and exploits popular Information Retrieval approaches, such as the *tf-idf* weighting scheme. In fact, the experimental evaluation on a benchmark dataset revealed a very promising performance of BSTR in terms of mAP and response time. However, the drawback of the proposed approach resides in the space occupation since it requires the expansion of the number of terms

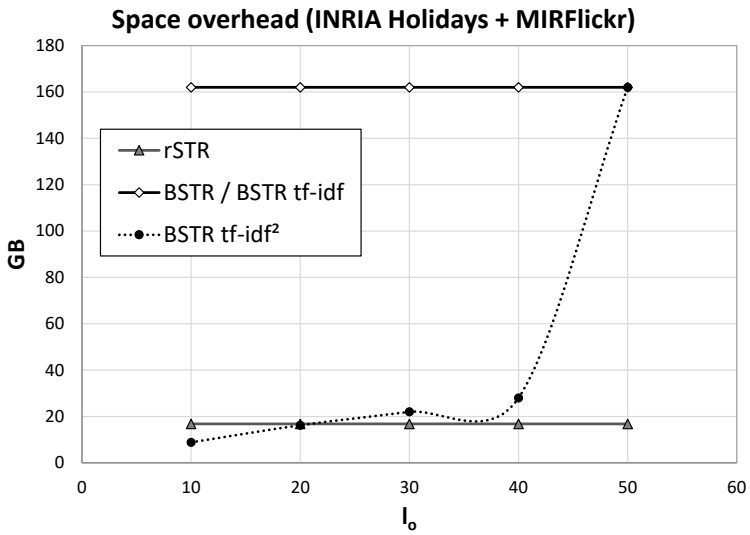


Figure 4.5: *Space occupation of the index for the different type of solutions, using the same value of $l_o = 50$ for BSTR and rSTR, and varying l_o for BSTR* tf-idf^2. *Note that for the rSTR, we consider also the overhead for the storage of the VLAD vectors used for the reordering of the results (higher values mean greater occupations).*

in the textual representation of the VLADs. This produces an inverted index that, using Lucene, is one order of magnitude greater than the baseline STR. To alleviate this problem, we propose to shrink the index by eliminating the terms associated with a low value of *tf*idf* (*i.e.* the BSTR *tf-idf*^{2} strategy). The last approach is efficient in term of both time and space overheads and still maintains good effectiveness. However, it requires a double indexing phase or at least a pre-analysis of the dataset in order to calculate the *tf-idf* weight of the terms. This analysis can be computed off-line and does not affect the cost at query time.

4.2 Deep Permutations

This section presents a novel approach to transform a deep feature into a permutation to be used in conjunction with a permutation-based index. As observed in Section 2.3, components of deep features are computed from neuron activations of a specific layer (usually one of the last layers) of a deep neural network. Typically they are sparse vectors with high dimensionality. For example, on INRIA Holidays dataset, the output of the sixth layer (*fc6*) of the well-known AlexNet [155] is a vector of 4,096 dimensions in which, on average, 75% of components have zero value after the ReLU activation function. In such cases, the exact search cannot deal with large collections of deep features while a more suitable choice is using approximate access methods. For example, PBI has been used in [28, 199] to index large volumes of these complex features so that similarity queries can be evaluated efficiently: Novak et al. [199] used PPP-Codes to index a collection of 20 million images processed by a CNN; Amato et al. [28] indexed 97M deep features using the MI-File [36]. In both cases, no special technique was used to generate permutations for deep features, which means that the permutations were computed on the basis of the proximity between a set of pivots and the data to be represented.

Notice that, the relevant aspect, when building permutations for PBI, is the capability of generating sequences of identifiers (permutations) in such a way that similar objects have similar permutations. Sorting a set of pivots, according to their distance to the object to be represented is just one, yet effective, approach. Here we propose to exploit the nature of the deep features in order to efficiently compute permutations directly from the feature components. Specifically, we propose to use the neurons as permutants and sort them according to their activation values. We named this technique *Deep Permutations*. The intuition behind our approach is that features in the high levels of the neural network carry-out

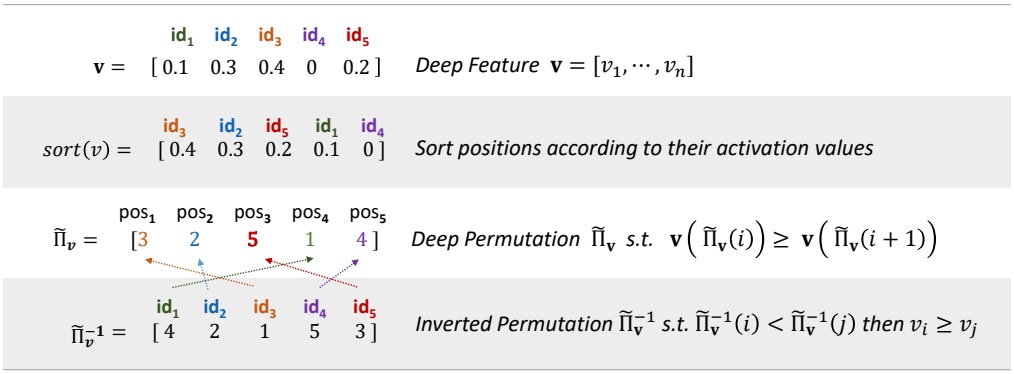


Figure 4.6: *Example of a Deep Permutation. The standard approach to generate a permutation is sorting pre-selected pivots according to their distance from the object to be represented. In the Deep Permutations, instead, the permutants are the neurons of the chosen layer (*i.e. *the dimension index of the deep feature), sorted according to their activation values.*

some high-level information.For example in the cases of a CNN feature extracted from an image, we can imagine that each neuron represents some sort of visual concept and its activation the importance of that visual concept in the image. Since each individual dimension of a deep feature corresponds to a neuron and the value of that dimension is the neuron activation, we have that similar images contain similar visual concepts and thus we expect they have similar Deep Permutations as well.

In the next sub-section, we discuss more in the details the process to compute the Deep Permutations. We then present experimental results, showing the effectiveness of our approach for both similarity search and retrieval tasks.

4.2.1 Computing the Deep Permutations

Given a deep feature $\mathbf{v} = [v_1, \ldots, v_n]$, we define the *Deep Permutation* $\tilde{\Pi}_\mathbf{v} = [\tilde{\Pi}_\mathbf{v}(1), \ldots, \tilde{\Pi}_\mathbf{v}(n)]$ as the ordering of the dimension indexes $\{1, \ldots, n\}$ such that

$$\forall i = 1, \cdots, n-1, \quad \mathbf{v}(\tilde{\Pi}_\mathbf{v}(i)) \geq \mathbf{v}(\tilde{\Pi}_\mathbf{v}(i+1)), \tag{4.13}$$

where we use the notation $\mathbf{v}(j)$ to indicates the j-th element of $\mathbf{v}$, that is v_j. In other words, permutants are the *dimension indexes* of the deep feature vector, which are sorted in descending order with respect to the values of the corresponding elements. So if the index i of the vector appears before index j in the permutation $\tilde{\Pi}_\mathbf{v}$, then the value v_i is greater than or equal to v_j.

Using the inverted representation $\tilde{\Pi}_\mathbf{v}^{-1} = [\tilde{\Pi}_\mathbf{v}^{-1}(1), \ldots, \tilde{\Pi}_\mathbf{v}^{-1}(n)]$, we have that if $\tilde{\Pi}_\mathbf{v}^{-1}(i) < \tilde{\Pi}_\mathbf{v}^{-1}(j)$ then $v_i \geq v_j$.

Figure 4.6 illustrates an example starting from the vector $\mathbf{v} = [0.1, 0.3, 0.4, 0, 0.2]$ [4]. The permutation-based representation of $\mathbf{v}$ is $\tilde{\Pi}_\mathbf{v} = [3, 2, 5, 1, 4]$, that is permutant (index) 3 is in position 1, permutant 2 is in position 2, permutant 5 is in position 3, etc. The inverted representation is $\tilde{\Pi}_\mathbf{v}^{-1} = [4, 2, 1, 5, 3]$, that is permutant (index) 1 is in position 4, permutant 2 is in position 2, permutant 3 is in position 1, etc.

Note that when two elements of a deep feature have the same value, their positions in the permutation cannot be uniquely assigned. This is very rare for the non-zero elements since we are working with real values. However, the problem remains with the zero values, that are very frequent when using ReLU activation function. This means that we cannot assign a unique ordering of the zero elements of the deep features.

In order to face this problem, we define and compare two different strategies:

[4] In reality, the number of dimensions is 4,096 or more.

- The first strategy, which we call *zeros-to-l*, assigns all elements having a value equal to zero to position $l + 1$, where l is a location parameter and then use the top-l truncated permutation.
- The second strategy, which we call *no-ReLU*, does not use the ReLU activation function during the feature extraction so that negative values are not flattened to 0 and vector components with the same activation values occur very rarely.

If we restrict to the case of Euclidean distance it easy to see that our strategy of generating permutations is equivalent to the following permutation generation strategy:

- Create a set of n pivots such that the i-th reference object has 1 in dimension i and 0 in all other elements of the vectors, *i.e.*

$$\begin{aligned} \boldsymbol{p}_1 &= [1, 0 \ldots, 0] \\ \boldsymbol{p}_2 &= [0, 1 \ldots, 0] \\ &\vdots \\ \boldsymbol{p}_n &= [0, 0 \ldots, 1] \end{aligned} \tag{4.14}$$

- Given an object $\mathbf{v}$ compute the traditional permutation $\Pi_{\mathbf{v}}$ that sorts all pivots in ascending order to their ℓ_2 distance from $\mathbf{v}$.

In facts, whenever $v_i \neq v_j$ we have that $\Pi_{\mathbf{v}}^{-1}(i) < \Pi_{\mathbf{v}}^{-1}(j)$ if, and only if, $\tilde{\Pi}_{\mathbf{v}}^{-1}(i) < \tilde{\Pi}_{\mathbf{v}}^{-1}(j)$. This is a direct consequence of the use of the Euclidean distance for the feature comparison, since $\ell_2(\mathbf{v}, \boldsymbol{p}_i)^2 = \|\mathbf{v}\|_2^2 + 1 - 2v_i$ and thus $\ell_2(\mathbf{v}, \boldsymbol{p}_i) < \ell_2(\mathbf{v}, \boldsymbol{p}_j)$ if, and only if, $v_i > v_j$.

Please note that in the Deep Permutations the number of permutants is the same of the dimension of the deep feature. A question that might arise is what is the benefit of using the Deep Permutation instead of the original vector given that they have the same dimensionality. The advantage of the proposed approach is that permutation vectors can be easily indexed, for example by encoding into an inverted index, which exhibits high efficiency as shown by Amato et al. [28, 36].

4.2.2 Experiments

As happens with the traditional permutation-based approach, searching using the Deep Permutations provides a result set that approximates the result of the exact similarity search. The desired situation is one in which we have a good approximation of the exact results as well as providing results that satisfy the user retrieval requirements. The latter means that the user does not notice the degradation of accuracy, given that also the exact similarity search algorithm is an approximation of his/her intuition of similarity.

In this respect, we performed two type of experiments. We first evaluated the performance of the proposed technique in a pure *similarity search task*, where we use an exact similarity search ground-truth to assess the quality of the approximate similarity search (Section 4.2.2.1). Then, we evaluated the performance in a *multimedia information retrieval task* (Section 4.2.2.2). Here, we used a ground-truth that was manually generated by associating each query with a set of results pertinent to the query. In this way, we were able to evaluate both the approximation introduced with respect to the exact similarity search algorithm and the impact of this approximation with respect to the user perception of the retrieval task.

4.2.2.1 Evaluation in a Similarity Search Task

In this first set of experiments, we evaluate how well the similarity between Deep Permutations reflects the similarity between the original feature vectors.

We used the *HybridNet fc6* features[5] extracted by Amato et al. [28] from the *Yahoo Flickr Creative Commons 100 Million dataset (YFCC100M)* [243]. The used feature are publicly available at `http://www.deepfeatures.org/`. Amato et al. also provide a ground-truth built selecting 1,000 different queries and executing an exact similarity search on these queries using the Euclidean distance to compare deep features. As previously noted, the activations at the *fc6* layer form a vector of 4,096

[5] We recall that the HybridNet [276] has the same architecture of the AlexNet but was trained on more images, as described in Section 3.1.2.3

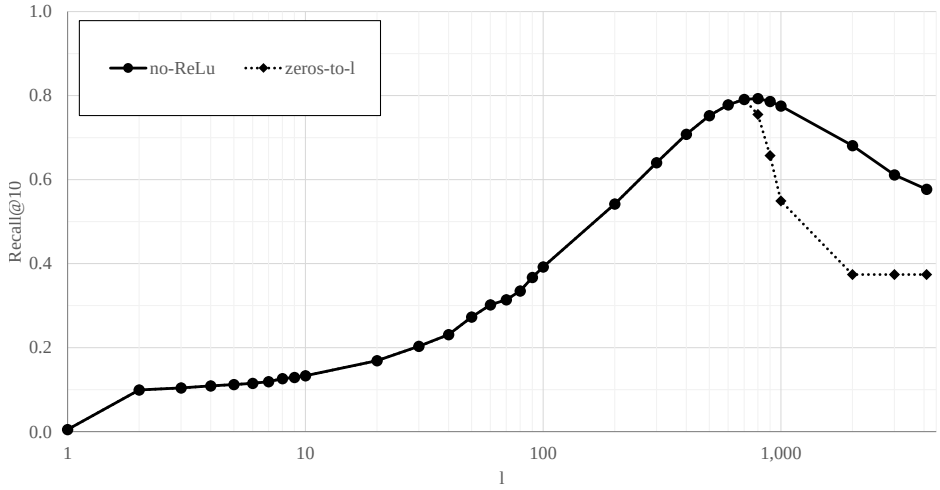


Figure 4.7: *Deep Permutations: comparison between the* no-ReLU *and the* zeros-to-l *techniques, varying the location parameter l (length of the truncated permutations).*

floats. Generally, the ReLU is used to bring to zero all negative activation values. In this way, feature vectors contain only values greater or equal to zero. The resulting feature vectors are quite sparse.

In order to assess the quality of search results, we use the *recall@k* (Eq (2.63), Section 2.4.9.1) that is the ratio between the number of correct results in the approximate result set and the number of correct results that should have been retrieved. In this context, we also compare the Deep Permutations with the traditional permutation representation.

no-ReLU *vs* zeros-to-l We first discuss the comparison of the two approaches that we defined for handling elements of the vectors having zero value, namely *no-ReLU* and the *zeros-to-l*. These tests were executed on a subset of the YFCC100M dataset of size 1M and the results are shown in Figure 4.7. In experiments, we evaluated the recall@10 varying the location parameter l of the truncated permutations (compared with the Spearman's rho with location parameter).

Note that the plots corresponding to the two strategies are overlapped until $l = 700$. Then, the *zeros-to-l* performance degrades with respect to the other. At $l = 900$ also the *no-ReLU* starts degrading, remaining always higher than the other. This behaviour is due to the presence of elements with a value equal to zero. In facts, for this dataset, we observed that on average the 75% of elements of *fc6* vectors are zeros, which cannot be uniquely sorted when computing the Deep Permutations. This means that when l approaches to 1,000, there are no more elements with non-zero values, which up to now were correctly sorted, and we encounter elements having a value equal to zero. The permutants (dimensional indexes) related to the zero value are assigned to position $l + 1$ when using the *zeros-to-l* approach, while all the zero values are replaced by the negative activation values, seen before applying the ReLU, in the *no-ReLU* approach. Our results show that using a negative value for sorting these elements helps, until a certain degree. It is worth mentioning that the features were extracted using a pretrained model and that the ReLU was used when the neural network was originally trained, so the negative values were always treated as zeros during the training. This means that the negative values were not subject of fine-tuning during the learning phase. This is the reason why we see a degradation also using the negative values in the *no-ReLU* strategy. It would be interesting comparing with a network trained using an activation function different from ReLU, as for instance the Exponential Linear Units (ELU) [70], that does not

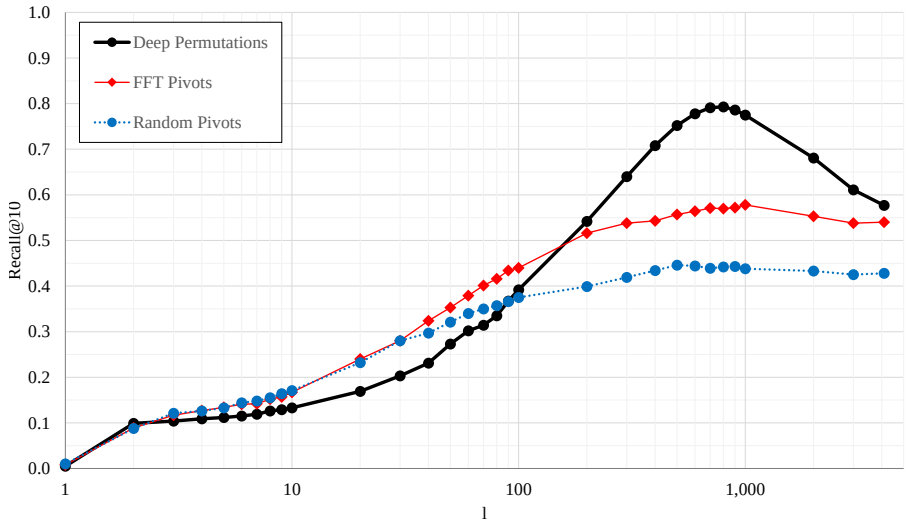


Figure 4.8: *Comparisons of the proposed Deep Permutation approach with standard permutation-based methods using random and Farthest-First Traversal (FFT) pivot selection strategies.*

flat the negative values. However, this was out of the scope of our investigation, where we wanted to use features extracted by standard CNNs by leveraging on transfer learning, as done in many other research work and applications.

Deep Permutations *vs* Standard Permutations Figure 4.8 compares the proposed Deep Permutations (using the *no-ReLU* strategy), with a standard permutation-based representation, which relies on distance calculations between a set of pivots and the data objects. For the latter approach, we considered two pivot selection techniques: random selection and FFT [83, 111]. In fact, while the random selection is one of the simplest and most used approaches, in [24] the FFT was identified as the best pivot selection method for permutation-based searching. Specifically, Amato et al. [24] compared the performance of several pivot selection strategies and the FFT provided a set of pivots such that the object ranking obtained using the similarity computed among the permutations was the most correlated to the ranking obtained using the original distance.

We notice that for values of l up to 200 the standard approach, both using Random selection and FFT offers better performance than Deep Permutations in terms of *recall*@10. Then, for values bigger than 200, the Deep Permutation approach performs much better, reaching a recall of 80%. The FFT approach is always lower than 60% and the random approach reaches just 45% recall@10.

We also compared the three approaches computing the $recall@k$ for various value of k ranging from 1 to 1,000. Results are showed in Figure 4.9. Also in this case, we can see that the new proposed approach outperforms the others and remains practically stable for all k values.

Tests discussed above were executed on a subset of size 1M of the entire YFCC100M dataset. Figure 4.10 shows the performance of the Deep Permutations approach increasing the size of the indexed dataset up to 100M. Here, we evaluated the recall for k equal to 10, 100, and 1,000. Also in this case we do not see significant differences for different values of k, and the recall remains also stable around 80% for the various tested sizes of the dataset.

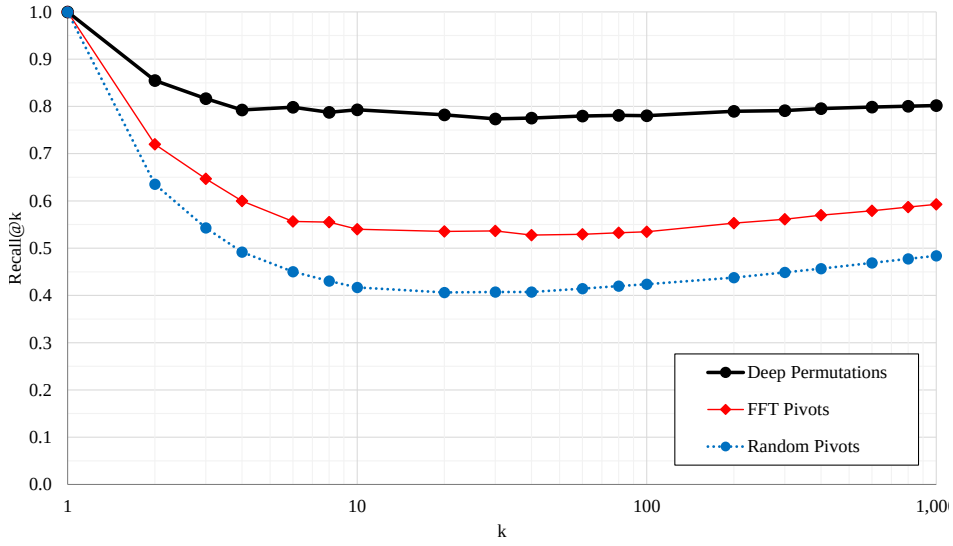


Figure 4.9: *Recall@k varying k for our approach with l=800 and 4,096 random and FFT pivots.*

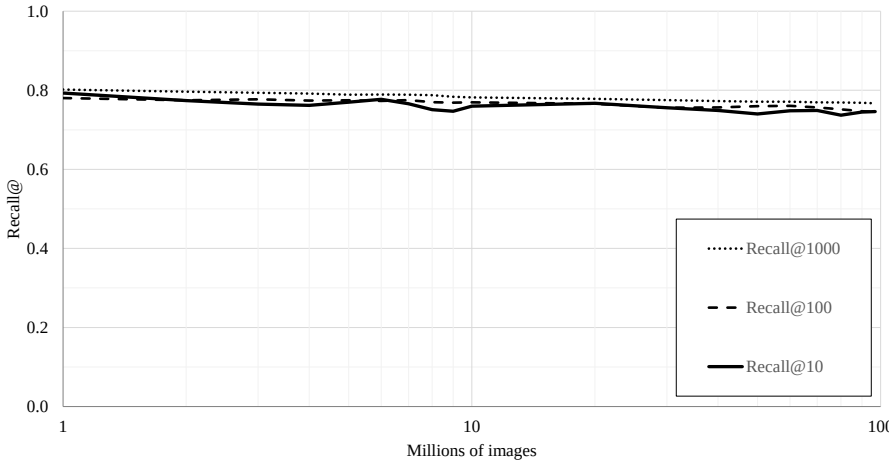


Figure 4.10: *Recall@k for various k varying dataset size (expressed in millions) obtained by the proposed approach for l=800.*

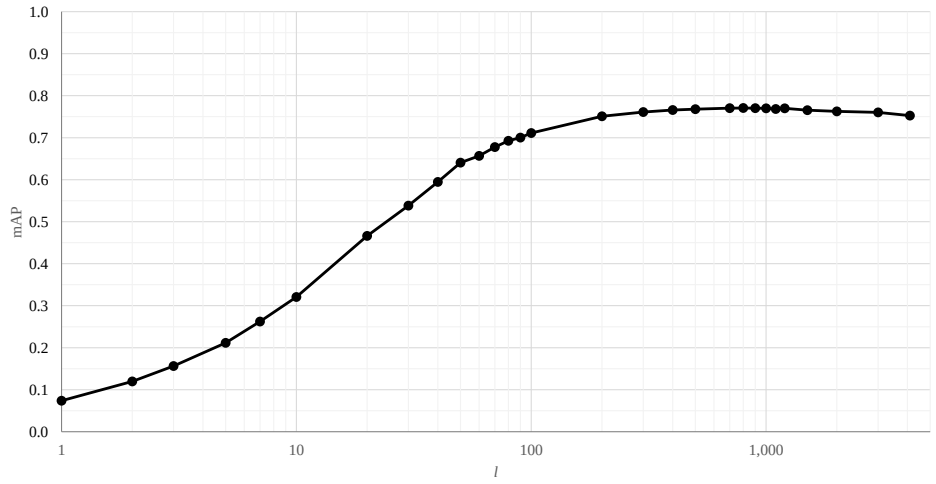


Figure 4.11: *mAP obtained on INRIA Holidays varying the location parameter l.*

4.2.2.2 Evaluation in a Multimedia Information Retrieval Task

Since deep features are widely used for image retrieval, we run a set of experiments to evaluate the performance of the proposed Deep Permutations in a multimedia information retrieval task. In this context, we need a manually-generated ground-truth to assess the various methods, so we used the *INRIA Holidays* [139] benchmark for which the ground-truth is publicly available. For large scale tests, we merged the Holidays dataset with the distraction dataset MIRFlickr [135] that contains 1M images. For both the datasets, the deep features were exacted taking the activation of the neurons in the *fc6* layer of the HybridNet. The retrieval performance is measured by means of mAP.

In these experiments, we only test the *no-ReLU* strategy since it provided us with the best results in the similarity search scenario.

Figure 4.11 shows the mean average precision on INRIA Holidays varying the length l of the truncated permutation. We can see that the *mAP* improves rapidly until l is 100, then remains stable slightly below 80%. The maximum is reached when $l = 800$, where the *mAP* is 77%. These values of *mAP* are rather surprising and competing with state-of-the-art methods tested on the INRIA Holidays dataset. To further investigate this we have compared the Deep Permutation with the direct use of the deep features, using the Euclidean distance, and with the recently proposed LuQ method [23]. The *LuQ* [23] uses a quantization process that allows one to use a text retrieval engine to perform image similarity search. Specifically, each real-valued vector component v_i of a deep feature is transformed in a natural numbers f_i given by $\lfloor Qv_i \rfloor$; where $\lfloor \rfloor$ denotes the floor function and Q is a multiplication factor > 1 that works as a *quantization factor*. The integer f_i are then used as term frequencies for the "term-components" of the text documents representing the feature vectors.

The comparison was performed on the INRIA Holidays dataset alone and together with the MIRFlickr dataset. Results are reported in Table 4.2. The direct use of the deep features on the INRIA Holidays dataset exhibits a mAP of 75% with *ReLU*, and 76% *without ReLU*. On the INRIA Holidays dataset with the MIRFlickr distraction set, it exhibits a mAP of 69% with *ReLU*, and 62% *without ReLU*. Our Deep Permutation on the INRIA Holidays dataset, achieves a mAP of 75% with full permutations, and 77% with $l = 800$. On the INRIA Holidays dataset with MIRFlickr distraction set, we obtain a mAP of 60% with full permutations, 62% with $l = 800$. The results obtained using $l = 800$ are always greater or equal to the one obtained directly using the deep features, and equal to the results obtained by LuQ.

Looking at these results we can make an additional observation. Deep features are generally compared using the ℓ_2 distance. However, results above suggest that possibly this is not the best distance function to be used. In fact, transforming deep features into permutations and comparing them using

Table 4.2: *Comparison of the mAP obtained on INRIA Holidays (with and without the MIRFlickr distraction set) using the following approaches: a) the direct use of the deep features compared with the Euclidean distance; b) our approach based on the use of the Deep Permutations compared with the Spearman rho distance; c) the LuQ method [23] which use the cosine similarity between text representations. For the first approach, we reported the results obtained with and without applying the ReLU during the extraction of the feature. For the Deep Permutations approach we reported the results obtained using the full length permutation and the truncated permutation with location parameter l=800*

	Deep features		Deep Permutations		LuQ [23]
	ReLu	no-ReLu	full	$l = 800$	
Holidays	0.75	0.76	0.75	0.77	0.77
Holidays+MIRFlickr	0.60	0.62	0.60	0.62	0.62

the Spearman rho distance has slightly better performance than the exact search. Thus, investigations on better distance functions to be used with deep features can be further considered. We performed some preliminary tests in this direction, using various vector transformations (ReLU, no-ReLU, ℓ_1 and ℓ_2 normalization, or even transforming the vector into a vector of probabilities) and testing various distances (ℓ_1, ℓ_2, Triangular Distance, Jensen-Shannon Distance, Bhattacharyya distance, Matusita distance and Pearson Distance) [6]. Surprisingly, for each used distance we observed that there exists at least one vector transformation for which the mAP on INRIA Holidays reaches a value between 0.75 and 0.76. However, we have not found yet a distance that works much better than the others.

4.2.3 Summary

This section presented the Deep Permutations approach that transforms deep features into permutations at very low computational cost.

Compared to the classical approach for computing the permutation-representation of an object, our technique does not need computing distances between pivots and the data object. Moreover, the proposed technique when evaluated in a pure similarity search task offered a recall much higher than other permutation-based representations. Furthermore, the Deep Permutations approach showed excellent performance also in a multimedia information retrieval context, where it reached a mean average precision slightly higher than the direct use of the deep features with the ℓ_2 distance.

At the time of writing, we notice that Hadjeres and Nielsen [119], inspired by our work, used Deep Permutations of features exacted from a RNN to encode symbolic musical sequences. In addition, Amato et al. [22] successfully used the deep permutations in conjunction with the R-MAC [113, 247] features and presently they are investigating novel activation functions to handle the negative values before computing the permutations.

[6] See [90] for definitions of Bhattacharyya, Matusita and Pearson distances

CHAPTER 5

Improving Supermetric Search through Finite Isometric Embeddings

In the previous Chapters, we have focused on efficient and effective approaches to extract and process image features. The next step to efficiently search a database of features is adopting appropriate indexing and searching algorithms. With this in mind, we have decided to focus on the metric search framework (introduced in Section 2.4) that thanks to its extensibility is used in many applications [66, 271], also including CBIR. In this context, the general interest is in searching a finite set of objects $\mathcal{S}$ that is a subset of a (infinite) domain $\mathcal{D}$ on which a metric function d is defined. Specifically, we want to efficiently find objects of $\mathcal{S}$ that are close to an arbitrary query object $q \in \mathcal{D}$, where the distance function is the only way by which two objects can be compared (the lower the distance the closer the objects). There are many query paradigms that can be employed, like "find the k closest objects to the query" (k-nearest neighbour query), or "find all the data objects within distance t of the query" (range query). The range query is perhaps the simplest one; moreover, it can be used as building block in the evaluation of more complex queries. For example, Hjaltason and Samet [132] proposed an algorithm to incrementally perform the k-nearest neighbour search by range queries with proper radius thresholds that can be found without computing additional distances.

In this Chapter, we examine a fundamental schema of metric search that is the exact search for range queries. Our choice may appear outmoded given that this topic was thoroughly inspected in past research literature. However, we have found out that some foundations of the metric search framework can be "reread" in the light of another consolidated theory that regards finite isometric embeddings in Euclidean spaces (introduced in Section 2.4.8.1). We have started our study from the following observations:

- Most of the existing partition-based indexes use the *triangle inequality* of the metric governing the space in order to estimate upper and lower bounds of the distances between the query and the data objects. The distance bounds are used to include or exclude partitions of the space from the search in order to avoid unnecessary distance calculations (space pruning, Section 2.4.6).

- The triangle inequality is equivalent to a discrete geometry condition, in facts, a semimetric space meets the triangle inequality (so it is a metric space) if, and only if, for any three points of the space there exists an embedding in a two-dimensional Euclidean space that preserves all the three inter-point distances. In other words, the space is isometrically 3-embeddable in ℓ_2^2 (Section 2.4.8.1).

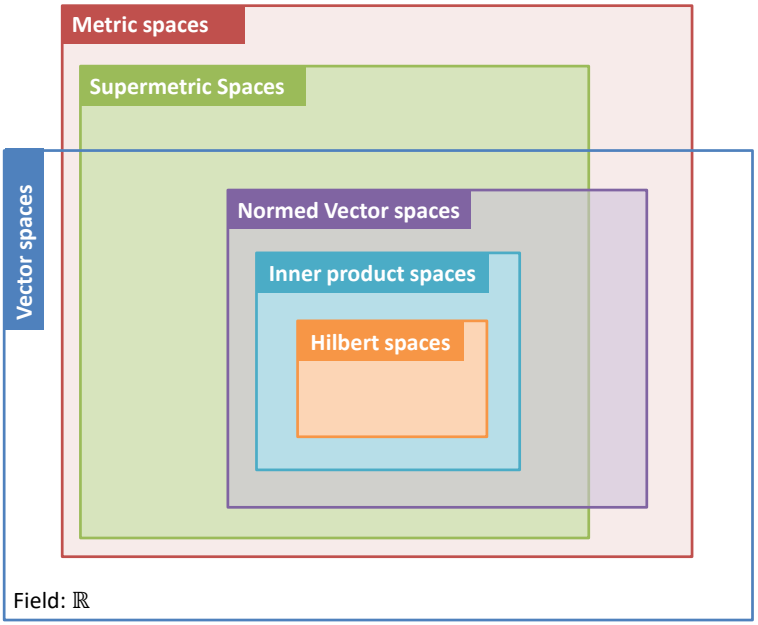


Figure 5.1: *Relations betweens metric spaces, supermetric spaces, and real vector spaces.*

- There exists a large class of metric spaces that have a stronger property, which is the isometric 4-embeddability in ℓ_2^3, *i.e.* for any four points of the space there exists an embedding into a three-dimensional Euclidean space that preserves all the $\binom{4}{2} = 6$ interpoint distances. This property is referred to as the *four-point property*.
- Many metric spaces that have the four-point property also meet the n*-point property*, that is for any n points of the space there exists an embedding into an $(n-1)$-dimensional Euclidean space that preserves all the $\binom{n}{2}$ interpoint distances.

These observations raise intriguing questions: is it possible to use the four (or n) point property to derive bounds on the distance that are tighter than those obtained using the triangle inequality? Is it possible to further exploit these geometric guaranties to speed up similarity queries?

We show that the answers to the above questions are affirmative whenever we work on spaces meeting the four- or n-points properties and we propose techniques that allow improving searching these spaces. Our approaches lose the universality of the metric search framework but still maintain a certain degree of his extensiveness: many of the metric spaces commonly used in applications meet the desired properties. In [76] we coined the term *supermetric* [1] *space* to refer to spaces with the four-point property as, in terms of metric search, they are significantly more tractable. Figure 5.1 illustrates the relations between the classes of supermetric spaces, metric spaces, and vector spaces; the former class is less universal than the metric one but is more general than the class of metric vector spaces.

By leveraging on the four-point property we derive a novel exclusion condition, named *Hilbert exclusion*, which results to prune more than the hyperbolic exclusion (Equation 2.54, Section 2.4.6), which is typically used in partition-based indexes, like hyperplane-based trees. This implies that a number of state-of-the-art indexing mechanisms over supermetric spaces can be easily refined to give a significant increase in performance. We further exploit the use of the n-point property to build a projection of the whole space into a n-dimensional Euclidean space, called *n-simplex projection*. We show how this projection can be exploited for indexing and dimensionality reduction on metric space satisfying the n-point property.

[1] This term has previously been used in the domains of particle physics and evolutionary biology as a pseudonym for the mathematical term *ultra-metric*, a concept of less interest in metric search; we believe our concept is of sufficient importance to the domain to justify its reuse with a different meaning.

This Chapter consists of four main sections. Section 5.1 gives a formal definition of our new exclusion condition and proves its applicability to any supermetric space. It also gives an analysis of the improvement of Hilbert exclusion over the hyperbolic exclusion, including relative performance measurements for some metric indexes that use hyperplane partitioning. Section 5.2 shows the four and n-point properties for important metric spaces, including spaces governed by Euclidean, Cosine, Jensen-Shannon and Triangular distances. Section 5.3 shows some stronger geometric guarantees deriving from the four-point property, which can be used to index the space. It also proposes a novel indexing structure (the Linear Regression tree) which is only possible to use in a supermetric space, as an example of the new area of investigation opened up by our research. Section 5.4 proves that the n-point property gives arbitrarily tight lower and upper bounds on distances between metric objects. Specifically, it presents a novel technique, called *n-simplex projection*, for embedding a large class of metric spaces into finite-dimensional Euclidean spaces. It also shows how this technique can be used to "compress" metric objects, similarly to other dimensionality reduction techniques, and to index and search the space.

The work presented in this Chapter has been jointly developed with the Prof. Richard Connor of the University of Strathclyde. Major results and findings were originally published in [74, 76–78]. The contributions of the author of this thesis has been in all the theoretical aspects, including, but not limited to, (i) setting the mathematical framework to prove the correctness of the Hilbert exclusion (ii) proving the applicability of our exclusion condition to a large set of metric spaces, (iii) proposing the Linear Regression tree, (iv) proving the correctness of the n-simplex projection and the related upper and lower bounds on the distances. Most of the experiments presented later were performed using codes originally implemented by Prof. Connor and publicly available at the *Metric Space Framework* [6].

Symbols and notation used throughout this Chapter are summarized in Table 5.1.

5.1 Hilbert Exclusion: a Novel Pruning Rule for Supermetric Spaces

Let $\mathcal{D}$ a domain and $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ a semi-metric on it, that is a function satisfying all the metric postulates except the triangle inequality. The semi-metric d also satisfies the triangle inequality if, and only if, the space $(\mathcal{D}, d)$ is isometrically 3-embeddable in $\ell_2^2 = (\mathbb{R}^2, \ell_2)$. Since most of the techniques used for metric indexing and space pruning are defined using the triangle inequality, it is interesting to express those properties in terms of the geometric guarantees afforded according to the 3-embeddability property in ℓ_2^2.

For example, let's consider a hyperplane-based partition of a finite search space $\mathcal{S} \subset \mathcal{D}$ according to two distinct pivots p_1 and p_2 (Section 2.4.5.3). We recall that this partitioning principle divides the space into two subsets according to which of the pivots is closer:

$$\mathcal{S}_{p_1} = \{o \in \mathcal{S} \,|\, d(o, p_1) \leq d(o, p_2)\}$$
$$\mathcal{S}_{p_2} = \{o \in \mathcal{S} \,|\, d(o, p_1) > d(o, p_2)\}.$$

The boundary between S_{p_1} and S_{p_2} is the hyperplane $H_{p_1,p_2} = \{o \in \mathcal{D} | \, d(o, p_1) = d(o, p_2)\}$. To evaluate a range query $R(q, t)$, the distances $d(q, p_1)$ and $d(q, p_2)$ are first calculated. If

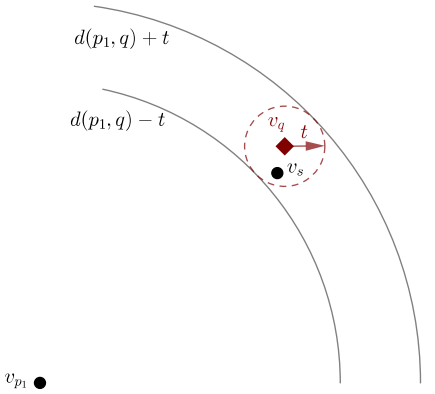
$$|d(q, p_1) - d(q, p_2)| > 2t \tag{5.1}$$

then the subset associated with the point further from q does not intersect with the solution set of the query and so it can be excluded from the search. The condition in Eq.(5.1) is the *hyperbolic exclusion*, which is a direct consequence of the double-pivot distance constraint (see Section 2.4.6.4). This exclusion condition is straightforward to derive algebraically from the triangle inequality property, but our main observation here is that it can also be shown in terms of 3-embeddability within ℓ_2^2. We have four actors: two pivots (p_1 and p_2), a query object q, and an arbitrary solution s (*i.e.* a point such that $d(s, q) \leq t$). If we consider an isometric embedding of p_1, q, and s in ℓ_2^2 that maps those objects into the 2D points $\boldsymbol{v}_{p_1}$, $\boldsymbol{v}_q$, and $\boldsymbol{v}_s$ (Figure 5.2a), we find out that the point $\boldsymbol{v}_s$ corresponding to the the solution s must lie in the region

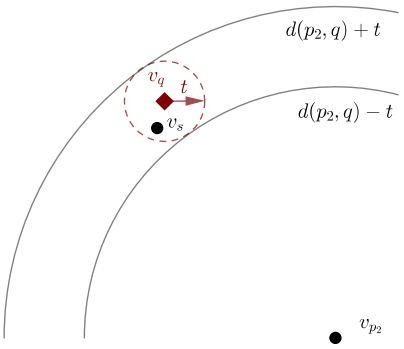
$$\{\mathbf{x} \in \mathbb{R}^2 \,|\, d(p_1, q) - t \leq \ell_2(\boldsymbol{v}_{p_1}, \mathbf{x}) \leq d(p_1, q) + t\}.$$

Table 5.1: *Notation and definitions used throughout this Chapter*

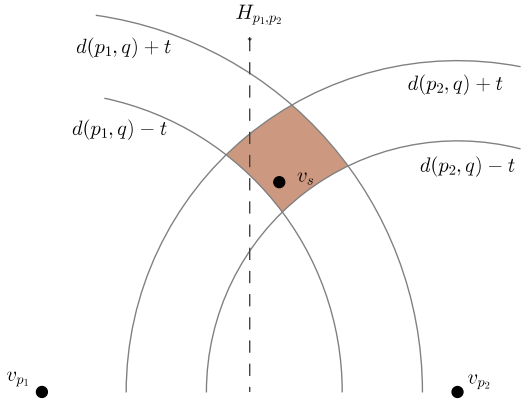
Notion	Definition
$(\mathcal{D}, d)$	Metric space: the data domain $\mathcal{D}$ and a metric distance d
$\mathcal{S}$	Finite set of data objects, $\mathcal{S} \subseteq \mathcal{D}$
m	Number of objects in S
s, o, s_i, o_i	Generic data objects in S
q	Query object $q \in \mathcal{D}$
t, t_i	Threshold distances used in the range search, $t, t_i \in \mathbb{R}$
$R(q, t)$	Solution set for a range query: $R(q,t) = \{o \in \mathcal{S} \mid d(o,q) \leq t\}$
$\{p_1, \ldots, p_n\}$	Set of n pivots, $p_i \in \mathcal{D}$
$\mathcal{S} = \mathcal{S}_{p_1} \cup \mathcal{S}_{p_2}$	Hyperplane partitioning of the space given two pivots p_1 and p_2: $\mathcal{S}_{p_1} = \{o \in \mathcal{S} \mid d(o,p_1) \leq d(o,p_2)\}$ $\mathcal{S}_{p_2} = \{o \in \mathcal{S} \mid d(o,p_1) \geq d(o,p_2)\}$
H_{p_1,p_2}	Hyperplane separating $\mathcal{S}_{p_1}$ and $\mathcal{S}_{p_2}$
Hyperbolic exclusion	$\lvert d(q,p_1) - d(q,p_2)\rvert > 2t$
Hilbert exclusion	$\frac{d(q,p_1)^2 - d(q,p_2)^2}{2\, d(p_1,p_2)} > t$ (introduced in Section 5.1)
ℓ_2	Euclidean distance: $\ell_2(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$, $x, y \in \mathbb{R}^n$
ℓ_2^n	n-dimensional Euclidean space: $\ell_2^n = (\mathbb{R}^n, \ell_2)$. For example, $\ell_2^2 = (\mathbb{R}^2, \ell_2)$, $\ell_2^3 = (\mathbb{R}^3, \ell_2)$
Isometric n-embedding in ℓ_2^{n-1}	A metric space $(\mathcal{D}, d)$ is isometrically n-embeddable in ℓ_2^{n-1} if for any n points $o_1, \ldots, o_n \in \mathcal{D}$ there exists a mapping function $f : \mathcal{D} \to \ell_2^{n-1}$ such that $\ell_2(f(o_i), f(o_j)) = d(o_i, o_j)$ for $i, j = 1, \ldots, n$
Four-point property	A metric space has the *four-point property* if it is isometrically 4-embeddable in ℓ_2^3
n-point property	A metric space has the *n-point property* if it is isometrically n-embeddable in ℓ_2^{n-1}
Supermetric space	A metric space meeting the four-point property
$\boldsymbol{v}_o$	Point in a Euclidean space corresponding to a data object $o \in \mathcal{D}$
$\overline{\boldsymbol{vw}}$	Edge between two point $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n$



(a) *isometric embedding of* p_1, s, q *in* ℓ_2^2



(b) *isometric embedding of* p_2, s, q *in* ℓ_2^2



(c) *isometric embedding of* p_1, p_2, s *in* ℓ_2^2

Figure 5.2: *Hyperbolic exclusion in the light of isometric 3-embedding in* ℓ_2^2*. Assume to have a generalized hyperplane partitioning of a metric space* $(\mathcal{D}, d)$ *given two pivots* p_1 *and* p_2*. The hyperplane* $H_{p_1,p_2} = \{x \in \mathcal{D} | \, d(x, p_1) = d(x, p_2)\}$ *divided the space into two subsets according to which of the two pivots is closer. The pivot* p_1*, the query* q*, and any solution* s *can be isometrically projected in two-dimensional Euclidean space (Fig.(a)). Similarly, we can isometrically project* p_2*,* q *and* s *in* ℓ_2^2 *(Fig.(b)). Finally, we can consider an isometric embedding of the two pivots and the solution* s *in* ℓ_2^2 *(Fig. (c)). Regarding the last embedding, the query point cannot be drawn in the same diagram, but given its distance from* p_1 *and* p_2*, we have that any solution in the original metric space must lie in the region bounded by the four arcs. If the whole of this region lies to the right (left) of the hyperplane* H_{p_1,p_2} *in* ℓ_2^2*, there is therefore no requirement to search to the left (right) of the hyperplane in the original space.*

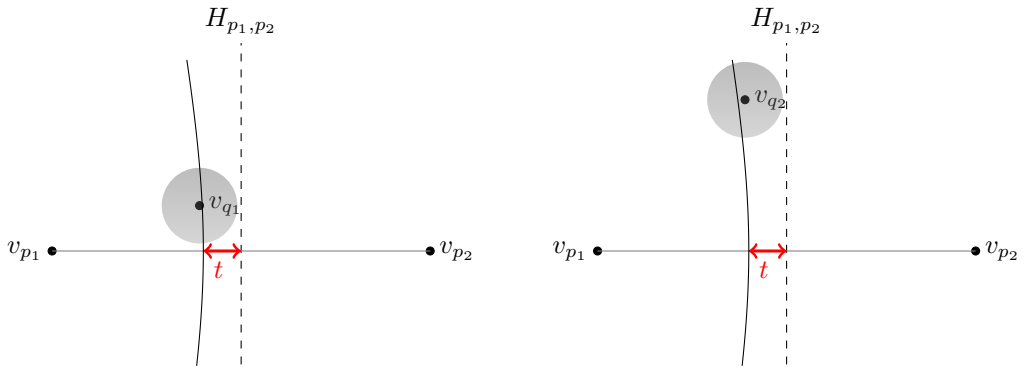


Figure 5.3: *Two queries, q_1 and q_2, each with threshold t, on the left side of the boundary H_{p_1,p_2}. Each object $o \in \mathcal{D}$ is represented by the 2D point $\boldsymbol{v}_o$ obtained using an isometric 3-embedding in ℓ_2^2 (in the depicted example, $\boldsymbol{v}_{p_1} = (-5, 0)$, $\boldsymbol{v}_{p_2} = (5, 0)$, $\boldsymbol{v}_{q_1} = (-1.1, 1.2)$, $\boldsymbol{v}_{q_2} = (-1.1, 4)$, $t = 1$). The hyperbola curve represents the embedding of all possible points $o \in S_{p_1}$ such that $d(o, p_2) - d(o, p_1) = 2t$,* i.e. *the boundary of the hyperbolic exclusion condition. The minimum distance of this hyperbola from the line H_{p_1,p_2} is t, but this occurs only on the line passing through $\boldsymbol{v}_{p_1}$ and $\boldsymbol{v}_{p_2}$. Since $\boldsymbol{v}_{q_1}$ lies on the left of the hyperbola - that is $d(q_1, p_2) - d(q_1, p_1) > 2t$) - the partition S_{p_2} can be excluded from the search on q_1. The query q_2, instead, is such that $d(q_2, p_2) - d(q_2, p_1) < 2t$ and so does not allow excluding half of the space. The circles of radius t plotted around the queries are meaningless with respect to the original metric space; we plotted them to visually perceive that in ℓ_2^2 the queries seems to have the same distance from the boundary line H_{p_1,p_2}, which gives the intuition that maybe also the solution set for the query q_2 is all contained in S_{p_1}. However, we have to inspect S_{p_2} to found it out.*

Similarly, if we consider the isometric embedding of p_2, q, and s in ℓ_2^2 (Figure 5.2b) we obtain that $\boldsymbol{v}_s$ lies in

$$\{\mathbf{x} \in \mathbb{R}^2 \mid d(p_2, q) - t \leq \ell_2(\boldsymbol{v}_{p_2}, \mathbf{x}) \leq d(p_2, q) + t\}.$$

Finally, let's consider the isometric embedding of p_1, p_2, and s in ℓ_2^2 (Figure 5.2c). In general, the point q cannot be isometrically embedded in the same plane as the two pivot points p_1, p_2, and the solution s, and therefore cannot be drawn in the same diagram. However, we know that for any isometric 3-embedding in a Euclidean plane that involves p_1, p_2, and s we must have

1. $d(p_1, q) - t \leq \ell_2(\boldsymbol{v}_{p_1}, \boldsymbol{v}_s) \leq d(p_1, q) + t$
2. $d(p_2, q) - t \leq \ell_2(\boldsymbol{v}_{p_2}, \boldsymbol{v}_s) \leq d(p_2, q) + t.$

This means that any solution s in the original metric space must lie in the region bounded by the four arcs shown in Figure 5.2c. Since the considered projection involves both p_1 ans p_2 we can also depict the hyperplane H_{p_1,p_2} in the same diagram, which corresponds to the line $\{\mathbf{x} \in \mathbb{R}^2 | \, \ell_2(\mathbf{x}, \boldsymbol{v}_{p_1}) = \ell_2(\mathbf{x}, \boldsymbol{v}_{p_2})\}$. In fact, we observe that for any two isometric 3-embeddings where two of the points are the same (in this case p_1 and p_2), then embedding functions can be chosen that map those two points to the same two points in ℓ_2^2, thus preserving the semantics of the line H_{p_1,p_2} (see also Section 5.1.2). The line corresponding to H_{p_1,p_2} divides the space into two subsets according to which of the two pivots is closer. Therefore if the whole of the region bounded by the four arcs lies to one side of this line, there is no requirement to search in the opposing side of the space. It can be seen from the diagram, that if q is closest to p_2, this occurs when $d(q, p_1) - t > d(q, p_2) + t$, that is $d(q, p_1) - d(q, p_2) > 2t$.

Figure 5.3 shows another example taken from a metric space that is 3-embedded in ℓ_2^2. It depicts the isometric embedding of three points (two pivots and a query) for two query alternatives. Of the two considered queries, only q_1 allows the partition on the far side of the hyperplane to be excluded, as for q_2 the exclusion condition is not met, even although the solution space "appears" geometrically separated from the right-hand side. This appearance, however, is an illusion: When considering this diagram in

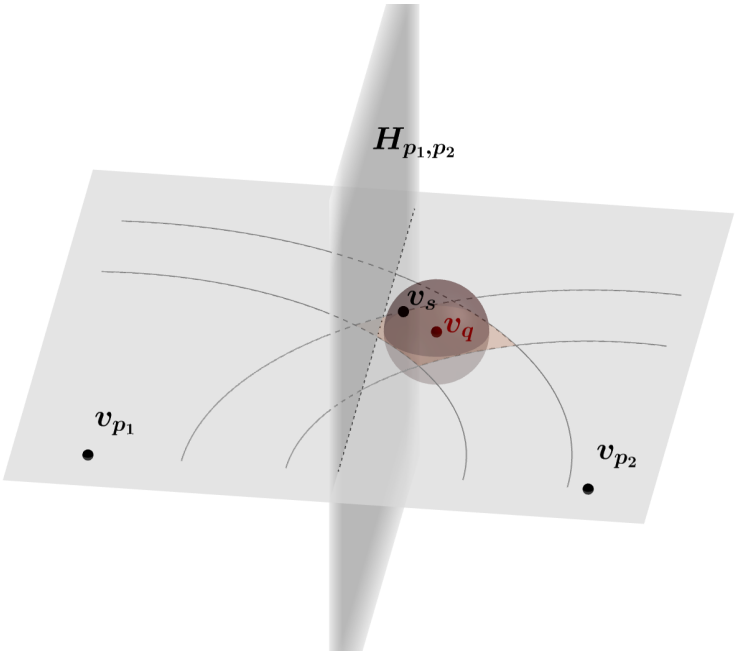


Figure 5.4: *Four points ($\boldsymbol{v}_{p_1}$, $\boldsymbol{v}_{p_2}$, $\boldsymbol{v}_q$, and $\boldsymbol{v}_s$) in ℓ_2^3,isometrically corresponding to four objects p_1, p_2, q and s, s.t. $d(q,s) \leq t$. For fixed p_1, p_2 and q, any solution to the query lies within the sphere centred around $\boldsymbol{v}_q$ in ℓ_2^3 and cannot belong to S_{p_1}, even although $d(q,p_1) - d(q,p_2) < 2t$. Note that H_{p_1,p_2} in the figure still represents the hyperplane that divides the space into two subspaces: objects nearer to p_1 belonging to the left subspace and objects nearer to p_2 to the right.*

two dimensions, the relative distances among $\boldsymbol{v}_{p_1}, \boldsymbol{v}_{p_2}$ and any individual $\boldsymbol{v}_{q_i}$ are significant, but as a general metric space guarantees only isometric 3-embeddability, the circles drawn around the 2D points corresponding to the queries are meaningless with respect to the original space.

Now we show that a tighter exclusion condition is possible if we consider spaces that are isometrically 4-embeddable in three-dimensional Euclidean space, that is spaces meeting the four-point property. Figure 5.4, shows the same situation of Figure 5.2c but relying on a 4-embeddability in ℓ_2^3. Here, the relative distances among any four points can be safely considered: in this case p_1, p_2, q, and any solution to q can be all drawn in ℓ_2^3. As before, we indicate with $\boldsymbol{v}_o$ the Euclidean point corresponding to the object $o \in D$. The plane on which the diagram is drawn is that containing $\boldsymbol{v}_{p_1}, \boldsymbol{v}_{p_2}$ and $\boldsymbol{v}_q$, and therefore the locus of any solution to q consists of a sphere in ℓ_2^3, radius t, centred around $\boldsymbol{v}_q$. It is clear from this diagram, in comparison with Figure 5.2c, that a more useful exclusion condition can be used: whenever the distance between $\boldsymbol{v}_q$ and H_{p_1,p_2} is greater than t, S_{p_1} does not require to be searched.

In the next subsection, we algebraically proves that this occurs when

$$\frac{d(q,p_1)^2 - d(q,p_2)^2}{2\,d(p_1,p_2)} > t. \tag{5.2}$$

We refer to this new exclusion condition to as *Hilbert exclusion*.

5.1.1 The Hilbert Exclusion Condition

Now we provide a formal definition and proof of correctness of Hilbert exclusion. We first consider the 3D Euclidean case, we then extend the proof for *supermetric spaces*.

Theorem 5.1.1. *Consider any three distinct points $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{q} \in \ell_2^3$ with $\ell_2(\boldsymbol{q},\boldsymbol{p}_1) > \ell_2(\boldsymbol{q},\boldsymbol{p}_2)$. Then the condition*

$$\frac{\ell_2(\boldsymbol{q},\boldsymbol{p}_1)^2 - \ell_2(\boldsymbol{q},\boldsymbol{p}_2)^2}{2\,\ell_2(\boldsymbol{p}_1,\boldsymbol{p}_2)} > t \tag{5.3}$$

implies that $\ell_2(\boldsymbol{s}, \boldsymbol{p}_1) > \ell_2(\boldsymbol{s}, \boldsymbol{p}_2)$ *for all* $\boldsymbol{s}$ *such that* $\ell_2(\boldsymbol{q}, \boldsymbol{s}) \leq t$.

Proof. It is sufficient to prove that the distance between the point $\boldsymbol{q}$ and the plane $H_{\boldsymbol{p}_1,\boldsymbol{p}_2} = \{\mathbf{x} \in \ell_2^3 \,|\, \ell_2(\mathbf{x}, \boldsymbol{p}_1) - \ell_2(\mathbf{x}, \boldsymbol{p}_2) = 0\}$ is greater than t. In this case, the ball with centre in $\boldsymbol{q}$ and radius t does not intersect the plane.

The equation of the plane $H_{\boldsymbol{p}_1,\boldsymbol{p}_2}$ can be rewritten as the scalar product $(\boldsymbol{p}_2 - \boldsymbol{p}_1) \cdot (\mathbf{x} - \frac{(\boldsymbol{p}_2 + \boldsymbol{p}_1)}{2}) = 0$, and so its distance from $\boldsymbol{q}$ is given by

$$\begin{aligned}
\text{dist}(\boldsymbol{q}, H_{\boldsymbol{p}_1,\boldsymbol{p}_2}) &= \left| \left(\boldsymbol{q} - \frac{(\boldsymbol{p}_2 + \boldsymbol{p}_1)}{2} \right) \cdot \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1)}{\|\boldsymbol{p}_2 - \boldsymbol{p}_1\|_2} \right| \\
&= \left| \frac{\|\boldsymbol{p}_1\|_2^2 - \|\boldsymbol{p}_2\|_2^2 + 2\boldsymbol{q} \cdot \boldsymbol{p}_2 - 2\boldsymbol{q} \cdot \boldsymbol{p}_1}{2\|(\boldsymbol{p}_2 - \boldsymbol{p}_1)\|_2} \right| \\
&= \left| \frac{\ell_2(\boldsymbol{q}, \boldsymbol{p}_1)^2 - \ell_2(\boldsymbol{q}, \boldsymbol{p}_2)^2}{2\,\ell_2(\boldsymbol{p}_1, \boldsymbol{p}_2)} \right|
\end{aligned}$$

Therefore, given that $\ell_2(\boldsymbol{q}, \boldsymbol{p}_1) > \ell_2(\boldsymbol{q}, \boldsymbol{p}_2)$, if $\text{dist}(\boldsymbol{q}, H_{\boldsymbol{p}_1,\boldsymbol{p}_2}) > t$, any point within distance t of $\boldsymbol{q}$ is closer to $\boldsymbol{p}_2$ than to $\boldsymbol{p}_1$ □

Theorem 5.1.2 (Hilbert Exclusion Condition)**.** *Let* $(\mathcal{D}, d)$ *a supermetric space and* $\mathcal{S} \subseteq \mathcal{D}$. *Given a range query* $R(q, t)$ *and two pivots* $p_1, p_2 \in \mathcal{D}$, *let* $\mathcal{S}_1 = \{o \in \mathcal{S} |\, d(o, p_1) \leq d(o, p_2)\}$, *and* $\mathcal{S}_2 = \{o \in \mathcal{S} |\, d(o, p_1) \geq d(o, p_2)\}$.

Assuming $d(q, p_2) < d(q, p_1)$, *then*

$$\frac{d(q, p_1)^2 - d(q, p_2)^2}{2\,d(p_1, p_2)} > t \quad \Rightarrow \quad \mathcal{S}_1 \textit{ can be excluded from the search.} \tag{5.4}$$

By symmetry, if $d(q, p_1) < d(q, p_2)$

$$\frac{d(q, p_2)^2 - d(q, p_1)^2}{2\,d(p_1, p_2)} > t \quad \Rightarrow \quad \mathcal{S}_2 \textit{ can be excluded from the search.} \tag{5.5}$$

Proof. Without loss of generality suppose that q is closer to p_2 than p_1. For any $s \in \mathcal{S}$ such that $d(q, s) \leq t$ we want to prove that $d(s, p_2) < d(s, p_1)$, that is any solution to the range query lie in the partition $\mathcal{S}_2$. Since $(\mathcal{D}, d)$ is isometrically 4-embeddable in ℓ_2^3, there exists a function $f : (\mathcal{D}, d) \to \ell_2^3$ which preserves all the six distances:

$$\ell_2\left(f(p_1), f(p_2)\right) = d(p_1, p_2) \tag{5.6}$$
$$\ell_2\left(f(q), f(p_1)\right) = d(q, p_1) \tag{5.7}$$
$$\ell_2\left(f(q), f(p_2)\right) = d(q, p_2) \tag{5.8}$$
$$\ell_2\left(f(s), f(q)\right) = d(s, q) \leq t \tag{5.9}$$
$$\ell_2\left(f(s), f(p_1)\right) = d(s, p_1) \tag{5.10}$$
$$\ell_2\left(f(s), f(p_2)\right) = d(s, p_2) \tag{5.11}$$

Equations (5.6)-(5.9) together with the condition in Equation (5.4) imply that points $\{f(p_1),\ f(p_2),\ f(q), f(s)\} \subseteq \ell_2^3$ satisfy the condition of the Theorem 5.1.1. It follows that $f(s)$ is closer to $f(p_2)$ than to $f(p_1)$:

$$\ell_2\left(f(s), f(p_1)\right) > \ell_2\left(f(s), f(p_2)\right).$$

This proves also that s is closer to p_2 than to p_1, in fact

$$d(s, p_1) = \ell_2\left(f(s), f(p_1)\right) > \ell_2\left(f(s), f(p_2)\right) = d(s, p_2).$$

□

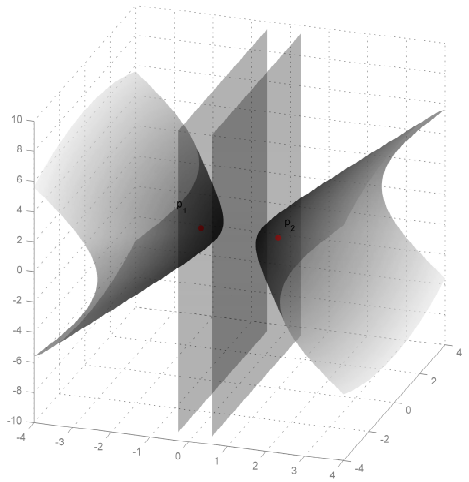


Figure 5.5: *Illustration of the Hilbert exclusion and hyperbolic exclusion in ℓ_2^3. Here pivots are placed at $(-1, 0, 0)$ and $(1, 0, 0)$, the threshold selected is 0.5. The planes and hyperboloids shown in the 3D space represent the boundaries of the Hilbert exclusion condition and of the hyperbolic exclusion condition respectively.*

Note that, for any solution s in $\mathcal{S}$, a different mapping function f may be required, however there is no requirement to identify it; the only importance of this function is that, for any four points, it exists.

The practical application of this theorem is in search indexes which partition the search space using two or more pivots: whenever the space has the four-point property, the Hilbert exclusion condition can be used in place of the hyperbolic exclusion condition. The important point in our context is that the first condition is weaker than the second, and is therefore a more useful exclusion condition.

Lemma 5.1.1. *The Hilbert exclusion condition is weaker than the hyperbolic exclusion condition.*

Proof. We require to prove that

$$\frac{d(q,p_1) - d(q,p_2)}{2} > t \quad \Rightarrow \quad \frac{d(q,p_1)^2 - d(q,p_2)^2}{2\,d(p_1,p_2)} > t$$

for which it is sufficient to show that

$$\frac{d(q,p_1)^2 - d(q,p_2)^2}{2\,d(p_1,p_2)} \geq \frac{d(q,p_1) - d(q,p_2)}{2}.$$

The last inequality directly follows from the triangle inequality:

$$\frac{d(q,p_1)^2 - d(q,p_2)^2}{2\,d(p_1,p_2)} = \frac{(d(q,p_1) - d(q,p_2))(d(q,p_1) + d(q,p_2))}{2\,d(p_1,p_2)} \geq \frac{d(q,p_1) - d(q,p_2)}{2}.$$

□

The proof of the previous Lemma also shows that the two conditions are the same only for a query point that is collinear with p_1 and p_2 in the isometric 4-embedding in ℓ_2^3; in all other cases the Hilbert exclusion is strictly weaker. In this context weaker implies better, as it allows more queries to exclude the opposing semispace from the search. Therefore, for the supermetric spaces more exclusion are always possible using the Hilbert exclusion rather than the hyperbolic one, and so in theory any indexing mechanism that uses the last one can be made more efficient [2]. Figure 5.5 gives an illustration of the two boundary conditions in a three-dimensional Euclidean space.

[2]The Hilbert exclusion uses the distance $d(p_1, p_2)$ other than $d(q, p_1)$ and $d(q, p_2)$. However, the distance $d(p_1, p_2)$ can be evaluated as the index is built, not during the query evaluation.

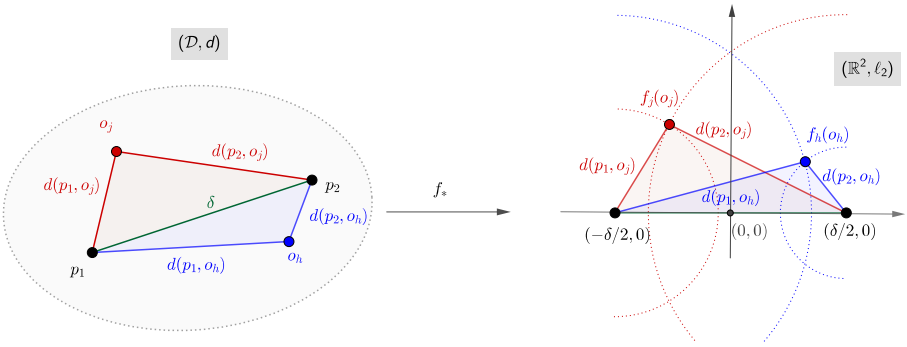


Figure 5.6: *Projection of two metric objects in 2D Euclidean space whilst preserving the distances between each point and two pivots.*

At the time of writing we notice that a formula equivalent to the left side of Equation (5.4) was used in the context of metric search also in [183, 214] to calculate the distance between the point q and the hyperplane equidistant from p_1 and p_2. This formula was derived using the cosine law on metric space with the "semidefinite positive property" [214, 229], since this property allows defining a notion of "angle" in a generic metric space. To provide a bridge to our work, we observe that the semidefinite positive property is equivalent to the *n-point* property for a finite semimetric space (see Chapter IV, Section 43 of [51]). Now we observe that this formula can be safely used in any space that satisfies the four-point property. While it is always true that the n-point property implies the four-point property, we cannot assert the vice versa. However, apart from pathological examples, we have not yet found a practical example of supermetric space that does not meet also the n-point property.

In the Section 5.2 we will prove that the n-point property is satisfied by a number of important metric spaces, notable including Euclidean spaces of any dimension, as well as probability spaces with the Jenson-Shannon and Triangular distances. Before giving a formal proof of the applicability of the Hilbert exclusion on these metric spaces, we would show the impact of using Hilbert rather than hyperbolic exclusion as well as present a first set of results obtained using our exclusion condition.

5.1.2 Exclusion conditions: 2D Visualization

The impact of a given exclusion condition over a set of queries can be easily visualized using 2D scatter plots, as described below.

Suppose to have a set of m object $\{o_1, \ldots, o_m\}$ and two pivots p_1, p_2 in a generic metric space (D, d). We know that for each triple (p_1, p_2, o_j) there exists an isometric embedding $f_j : (D, d) \to \ell_2^2$ that preserves all the three interpoint distances. Without loss of generality, we can assume that all the f_j maps the pivots p_1 and p_2 to the X axis, either side of the origin, separated by the distance between them in the original space (*i.e.* $p_1 \mapsto (-\delta/2, 0)$ and $p_2 \mapsto (\delta/2, 0)$, where $\delta = d(p_1, p_2)$)[3]. Each point o_i ($o_j \neq p_1, p_2$) is mapped to a point $f_j(o_j) \in \mathbb{R}^2$ that preserves the distances $d(p_1, o_j)$ and $d(p_2, o_j)$. It is easy to verify that the coordinates of $f_j(o_j)$ are either (x_j, y_j) or $(x_j, -y_j)$, where

$$x_j = \frac{d(o_j, p_1)^2 - d(o_j, p_2)^2}{2d(p_1, p_2)} \tag{5.12}$$

$$y_j = \sqrt{\frac{d(o_j, p_1)^2 + d(o_j, p_2)^2}{2} - x_j^2 - d(p_1, p_2)^2/4} \tag{5.13}$$

For the sake of simplicity, let's consider only the case where all the points are projected in the upper half of the plane. Figure 5.6 exemplifies the projection for two objects.

Using this procedure we are able to "visualize" all the points in a 2D scatter plot, such that all the distances $d(p_i, o_j)$ are preserved for $i = 1, 2$ and $j = 1, \ldots, m$, as well as $d(p_1, p_2)$. However, the

[3] Another simple choice that we use later is $p_1 \mapsto (0, 0)$ and $p_2 \mapsto (\delta, 0)$.

distances $\ell_2(f_j(o_j), f_h(o_h))$ are of no significance at priori since each point is obtained using a different isometric embedding function.

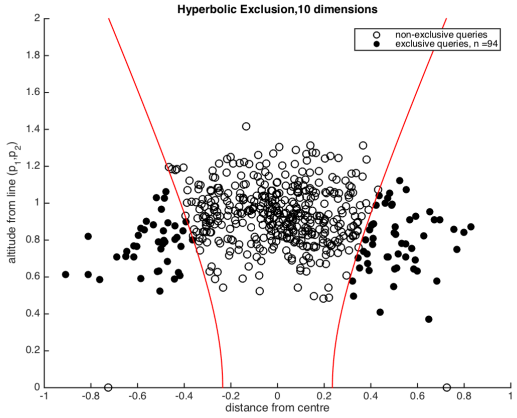
It is worth noting that the proposed visualization preserve the semantic of any locus of points whose conditions are defined in terms of the distance between a point and the pivots. So for example, the hyperplane $H_{p_1,p_2} = \{o \in \mathcal{D} \,|\, d(o,p_1) = d(o,p_2)\}$ will corresponds to the Y-axis, the boundary of the hyperbolic condition $\{o \in \mathcal{D} \,|\, d(o,p_1)^2 - d(o,p_2)^2 = 2t\}$ will correspond to the hyperbola $\{(x,y) \in \mathbb{R}^2 \,|\, (x+\delta/2)^2 - (y-\delta/2)^2 = 2t\}$, etc.

The mapping described above can be used to visualize the difference in pruning capability of Hilbert and hyperbolic exclusion conditions. Figure 5.7 shows an example in 10-dimensional Euclidean space. Each scatter plot shows the same set of 500 randomly generated points within the unit hypercube. A further two points p_1, p_2 are also generated to act as pivots and a query threshold t is chosen to return around one point per million from a large set. In the cases shown in Figure 5.7a and Figure 5.7b the objects are divided into two subsets according to which side of the Y-axis they lie (which correspond on a hyperplane partitioning of the original space). The solidly-coloured point are points that were they queries, would allow the semispace on the opposing side to be excluded from the search. We refer to these points as "exclusive queries". The other points are referred to as "non-exclusive queries". Figure 5.7a highlights the points that are exclusive queries according to the hyperbolic exclusion; Figure 5.7b highlights the points that are exclusive queries according to the Hilbert exclusion. To give reference diagram for single pivot-based exclusion, in Figure 5.7c we also consider a ball-partitioning defined according to the median distance to the left-hand pivot point. In this case, the exclusive queries are those that allow half of the space to be excluded according to the range-pivot pruning rule (Section 2.4.6.2). In this example, the number of exclusive queries is substantially greater for Hilbert exclusion (219 against 94 for hyperbolic exclusion and 137 for range-pivot exclusion). It is also instructive to note that this scatter plots clearly show the shape of the boundary conditions: hyperbola with foci at the reference points and semi-major axis equal to t for the hyperbolic exclusion, parallel lines at distance t from the separating hyperplane (the Y-axis) for the Hilbert exclusion, and concentric circles for the range-pivot case.

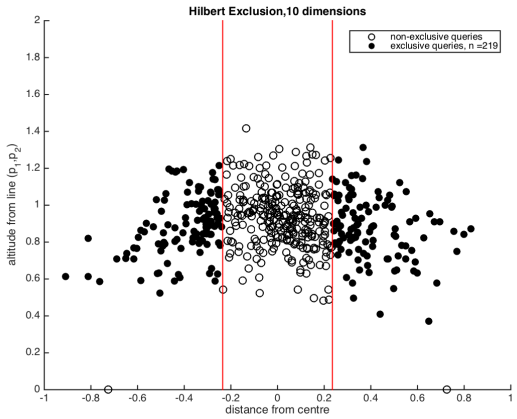
5.1.2.1 The Effect of the Reference Point Separation

An important observation is that the Hilbert condition, unlike the hyperbolic exclusion, is not affected by the separation of the reference points. In fact, the lines that bound the "exclusive queries" for the Hilbert exclusion are defined only by the query threshold t, since they are always parallel and at distance t from the hyperplane used to partition the space. The hyperbola which bounds the "exclusive queries" for the hyperbolic exclusion is defined by the query radius and the distance between the reference points, where the larger the separation of the reference points, the better the exclusion. If the reference points are too close each other, the hyperbola has very high curvature. In the extreme case where the separation is no larger than twice the query radius, which can readily occur in high-dimensional space, it is impossible for any exclusions to be made. This effect can be ameliorated by choosing widely separated reference points since in this case the hyperbola curvature is low and the query is more likely to be "exclusive". However, having more exclusive queries, in this case, does not guarantee more exclusion with respect to a less wide choice of the reference points. For example, in an unevenly distributed dataset if one of the reference point chosen is an outlier, then the point cloud will lie close to the other point, and again no exclusions will be made. Finding two reference points which are well separated, and where the rest of the points is evenly distributed between them, is an intractable task in general.

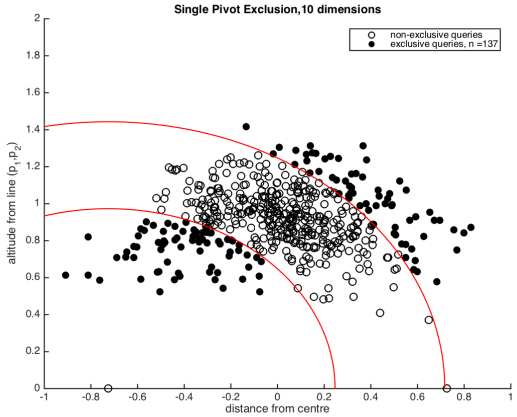
Figures 5.8 and 5.9 show the effect of the separation of the reference points. We consider a set of 500 randomly generated points in 8-dimensional Euclidean space. In these diagrams, the reference points have been selected as the furthest, and nearest, respectively out of 1,000 sample pairs of points drawn from the space. It can be seen that, when the exclusion is based on the four-point property (Hilbert exclusion), the exclusive power remains fairly constant. However, when the hyperbolic condition is used, the exclusive power is hugely affected; in this case, the query threshold is only slightly less than half the separation of the reference points, and the resulting hyperbola diverges so rapidly from the separating hyperplane that no exclusions are made from the sample queries. From Figure 5.8 it should also be noted that, no matter how far the reference points are separated, the four-point property always gives a higher



(a) *Hyperplane partitioning + hyperbolic exclusion*



(b) *Hyperplane partitioning + Hilbert exclusion*



(c) *Ball partitioning + Range-Pivot exclusion*

Figure 5.7: *Exclusion capability: Comparison between hyperbolic exclusion, Hilbert exclusion and range-pivot exclusion for 500 randomly generated points in 10-dimensional Euclidean space. The points are partitioned in the original space using Hyperplane partitioning ((a) and (b)), or using Ball partitioning according to the median distance from the left-hand pivot (c). For the hyperplane partitioning cases, the points coloured solidly are those which, were they queries, would allow the opposing semispace to be excluded from a search. For the ball partitioning case, the points coloured solidly indicate those whose distance from the pivot point is more than the query threshold away from its median.*

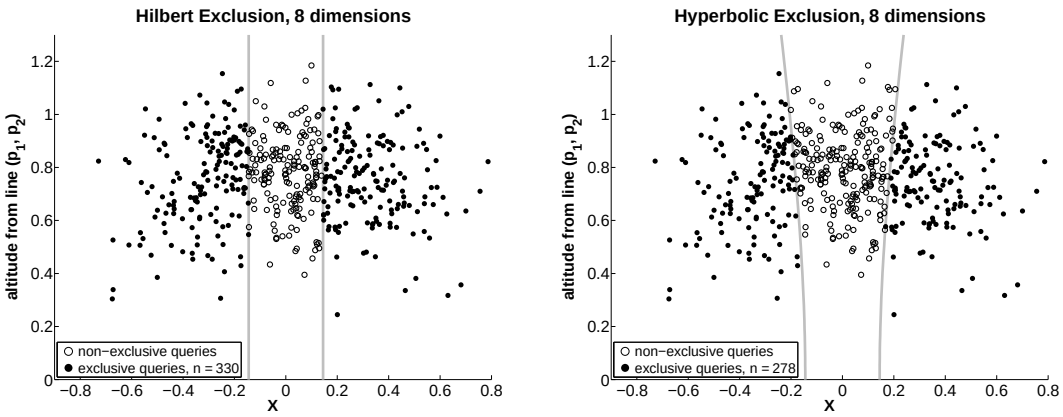


Figure 5.8: *Scatter diagram for 8-dimensional Euclidean space with widely separated reference points. (The distance between reference points is such that the reference points themselves do not appear on the plot).*

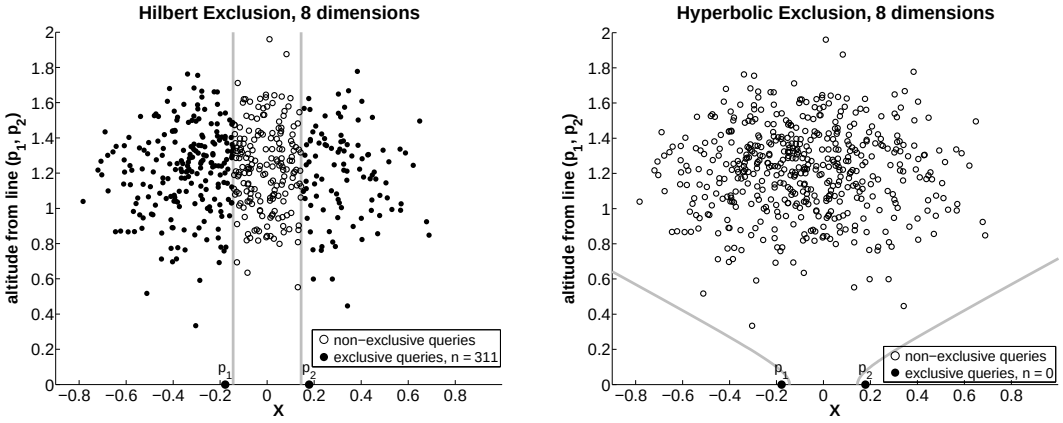


Figure 5.9: *Scatter diagram for 8-dimensional Euclidean space with close reference points. Note from the comparison of the left-hand graphs of this figure with Figure 5.8 that the separation of the reference points has no apparent effect on the power of the Hilbert exclusion, whereas normal hyperbolic exclusion becomes completely useless.*

probability of exclusions; in this case, although the separating lines do not appear visually to be very different, the implied probability of exclusion in for the Hilbert condition is 0.66, against 0.56 for the hyperbolic condition.

5.1.3 Experimental Evaluation

Given that the Hilbert exclusion condition is strictly weaker than the hyperbolic one (Lemma 5.1.1), we expect that any partition-based indexing mechanism will always perform better if using the former condition. Both the conditions require the evaluation of the distance between the query and each pivot. The Hilbert exclusion also requires the distance between the two pivot points, however, this may always be calculated during the building of any indexing structure. The minor increase in arithmetic cost and the extra space required to store the distance between pivots would not normally make a significant difference to the query cost given that this cost is totally dominated by the number of dynamic distance calculations required and the use of memory where the objects are large.

We are aware that is not possible to give a simple measure of “performance improvement" of our exclusion condition in general terms, since performance is highly dependent on many factors (*e.g.*, the cost of the distance calculations, the size of the objects, the intrinsic dimensionality of the space, the used

index mechanism, etc.). We, therefore, give an analysis of the improvement of our exclusion condition over the hyperbolic exclusion in two fundamental scenarios as follows.

Tests on synthetic data (Section 5.1.3.1) We consider several synthetic datasets generated for various dimensions and metrics. For each space, we first measure the ability of both Hilbert and hyperbolic mechanisms to avoid searching either half of the space without using any specific index structure. We then measure the improvement in term of distance calculations obtained using our exclusion condition in conjunction with data structures relying primarily on hyperplane partitioning (hyperplane and bisector trees).

Tests on "real-world" data (Section 5.1.3.2) The "Similarity Search and Applications" (SISAP) forum[4] publishes a number of large datasets drawn from real-world contexts which are commonly used as benchmarks for metric indexing approaches. Results over these have been reported for many different indexing mechanisms. We focus on exact search and take the best of these mechanisms at the time of writing that is DiSAT. DiSAT uses both radius and hyperplane exclusion mechanisms. We compare the results obtained with DiSAT using either Hilbert or hyperbolic condition to perform the hyperplane-based exclusion.

All of our measurements are expressed in terms of the number of distances calculation, and we do not give any measured execution times. The reason is that our focus here is to compare two exclusion mechanisms, so the number of distance calculations is the more important outcome in this context. In fact, our measured times for experiments we perform are approximately proportional, as the search structures are built over relatively small datasets which fit wholly within main memory and the cost of distance calculations is dominant.

5.1.3.1 Tests on Synthetic Data

We start our analysis by testing Hilbert and hyperbolic exclusion on a variety of spaces and search thresholds. We use pseudo-random datasets of one million elements within the unit hypercube, uniformly distributed in each dimension, within $\mathbb{R}^D$ for $D \in \{6, 8, 10, 12, 14\}$. We considered three different metrics over these spaces: Euclidean, Jensen-Shannon and Triangular distances. For Jensen-Shannon and Triangular distances, each point was normalized so that the sum of all entries is 1.

We evaluate the performance of the two hyperplane-based exclusion mechanisms in term of

- *Exclusion power:* for a given finite space, we randomly select pairs of reference points that partition a space into two halves (balanced hyperplane partitioning). We then measure the "exclusion power" of each exclusion mechanism as the probability of a randomly-selected query being able to avoid searching either half of the space based only on its distance from the two points.
- *Improvement over hyperplane-based tree:* for a given metric space, we build simple data structures relying primarily on hyperplane partitioning, namely Generalized Hyperplane Tree (GHT), Bisector Tree (BST) and Monotonous Bisector Tree (MBT). The same index structures can be used with either Hilbert or hyperbolic exclusion; improvement is measured as a simple multiplicative factor between the number of distance calculations for the two exclusion mechanisms.

In the results presented we name the spaces used based on the metric and the number of Cartesian dimensions, *e.g.* `euc_8` for Euclidean distance over $\mathbb{R}^8$, `jsd_10` for Jensen-Shannon distance over $\mathbb{R}^{10}$ etc. Search thresholds were derived by experiment, for each space, as those which would return around k results per million data, for $k \in \{1, 4, 16\}$. For each space, we also report the Intrinsic Dimensionality (IDim) [66] generally believed to give a reasonably good estimate of the tractability of a space to metric indexing techniques (see Section 2.4.2). A common observation is that spaces with an IDim of greater than around 6 are challenging, and those with an IDim of greater than about 10 are usually intractable with exact search. We estimate the IDim as in [66], which is defined over a sample of distances calculated over randomly selected points from within the space, as $\mu^2/(2\sigma^2)$ where μ is the mean and σ is the standard deviation of these distances. Table 5.2 gives values of IDim and search thresholds calculated for each space.

[4] http://www.sisap.org

Dataset	IDim	Thresholds		
		t_1	t_4	t_{16}
euc_6	7.64	0.076	0.095	0.120
euc_8	10.5	0.149	0.177	0.211
euc_10	13.3	0.228	0.262	0.301
euc_12	16.1	0.308	0.346	0.388
euc_14	19.0	0.386	0.426	0.471
jsd_6	5.15	0.022	0.030	0.040
jsd_8	7.26	0.045	0.057	0.071
jsd_10	9.39	0.067	0.079	0.094
jsd_12	11.4	0.084	0.099	0.114
jsd_14	13.7	0.103	0.118	0.133
tri_6	5.76	0.025	0.035	0.047
tri_8	8.25	0.053	0.068	0.083
tri_10	10.6	0.078	0.093	0.110
tri_12	13.0	0.098	0.116	0.133
tri_14	15.5	0.120	0.137	0.155

Table 5.2: *Synthetic datasets for various metrics (Euclidean, Jensen-Shannon and Triangular distances), and number D of Cartesian components ($D \in \{6, 8, 10, 12, 14\}$). For each metric space, threshold t_k was experimentally derived as that which would return around k results per million data.*

All results presented in the following are independent of the computer upon which they are performed, and all figures presented represent mean values where experiments were repeated until the standard error of the mean was less than 1% of the value given.

Exclusion Power Test Here we compared the exclusion power of Hilbert and hyperbolic conditions in the most basic scenario that is without using any index structure. For reference we also considered the results related to range-pivot exclusion; however, the fair comparison in this context is between the Hilbert and the hyperbolic exclusion since they are based on the same partitioning principle and they are mutually exclusive. The range-pivot exclusion, instead, is normally used to effect in conjunction with hyperplane-based exclusion and so it can be used with either Hilbert or hyperbolic condition. Moreover, the range-pivot condition requires one distance calculation, while hyperplane-based exclusion, in theory, requires two distance calculations. In practice, many indexes (like monotone trees) have ways of amortising this extra cost.

Results are reported in Figure 5.10. In each test, we partitioned the space in two halves with respect to some pivots and we measure the probability of a randomly-selected query being able to avoid searching either half of the space based only on its distance from the pivot(s). The left-hand figures show the exclusion percentage obtained at various dimensions and thresholds for Euclidean, Jenson-Shannon and Triangular distances. In all spaces that we have measured the single range-pivot method has more exclusion power than hyperbolic exclusion, but less power than Hilbert exclusion. The results have similar trends for all the tested metrics. So the efficiency of Hilbert exclusion does not seem to be affected by the choice of the metric. Moreover, it can be seen that Hilbert exclusion performs much better than hyperbolic exclusion, and is much more tolerant to increases in both dimensionality and query threshold; that is, it performs relatively better as the space becomes less tractable.

The right-hand graphs illustrate this in terms of improvement of Hilbert over hyperbolic exclusion, which again can be seen to increase sharply as space becomes less tractable.

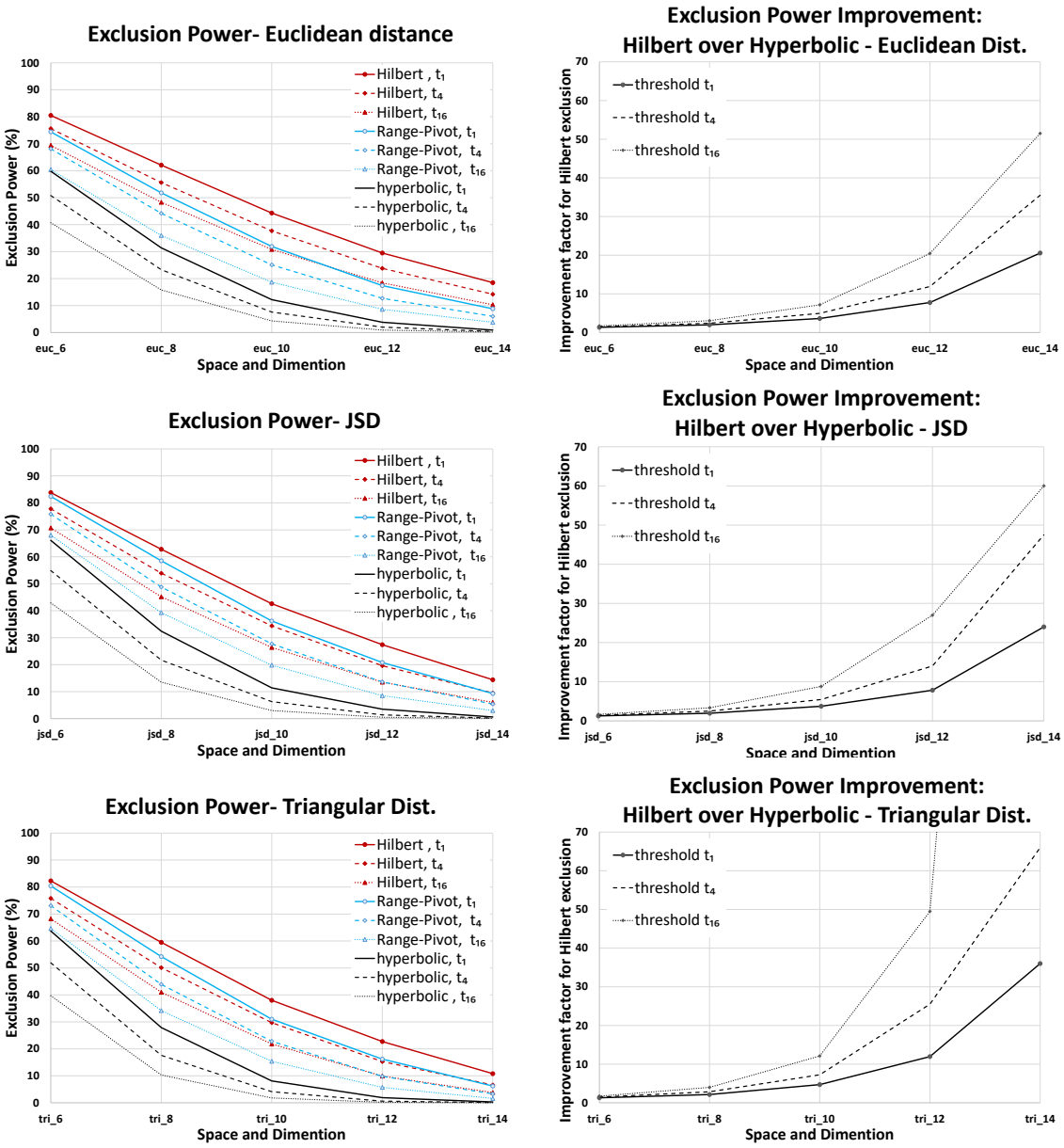


Figure 5.10: *Exclusion Power tests: Each figure shows five different dimensionalities, and three different search thresholds, for Euclidean distance (top row), Jensen Shannon distance (middle row), and Triangular Distance (bottom row). Left figures are percentage exclusion, right figures are the relative improvement of Hilbert over hyperbolic.*

Improvement tested over hyperplane-based trees To give a more practical measurement of performance improvement of Hilbert over hyperbolic, we tested the two exclusion mechanisms using metric indexes over the previously described synthetic datasets. The indexes used are the Generalized Hyperplane Tree (GHT) and the Bisector Tree (BST) which are in a sense the most "pure" (and certainly the simplest) hyperplane indexing structures. We also tested the monotone version of the bisector tree, namely the Monotonous Bisector Tree (MBT) [5]. We remind that GHT and BST share the quite same construction algorithms: the only difference is that the "bisector" tree stores the covering radius of each sub-partition so that both pivot-based exclusion and hyperplane exclusion are used at query time. The monotone version of the bisector tree, *i.e.* MBT, re-uses pivots of the internal nodes so that fewer distance computations are needed to build the index and execute a query. In our experiments, we used both cover radius and hyperplane exclusion mechanisms when evaluating bisector trees, as would be normal in practice, and compared the use of hyperbolic exclusion with Hilbert exclusion. At each node we used the "Far" pivot selection strategy, that is: the first pivot is chosen either randomly or handed down from an ancestor node for the monotone case, the second pivot is selected within the data subset used to construct that node so that it is at the furthest distance from the first pivot.

Table 5.4 reports the results for Euclidean, Jensen-Shannon and Triangular distances for five different dimensions $(6, 8, 10, 12, 14)$. Figure 5.11 shows the results for the Euclidean case in graphical form. Results for Jensen-Shannon and Triangular metrics have similar trends.

It can be seen that, for all spaces, Hilbert condition always gives better performance than hyperbolic condition; this is expected, as the former exclusion condition is strictly weaker. The bisector trees (BST, MBT) always give improved performance over the GHT, also thanks to the joint use of pivot-based and hyperplane-based exclusions. The improvement given by using Hilbert exclusion with the GHT is impressive (up to six times fewer distance calculations in 10D Euclidean space). It can also be seen that the GHT under Hilbert exclusion gives similar performance than the BST under both hyperbolic exclusion and range-query exclusion. Interestingly, the improvement given by using Hilbert exclusion with the MBT is better than the improvement given over the BST. This is better shown in Figure 5.12, where results for Euclidean spaces are interpreted as improvement ratio. We also observed that for all search thresholds, the improvement ratio reaches a maximum at around 10 dimensions and then decreases again. This can be explained by the fact that, for very tractable spaces, both mechanisms function very well; there is not, therefore, a great improvement. For intractable spaces, neither mechanism can do well and so again the relative improvement becomes less. So, it can be seen that the gap in the exclusion power of the two mechanisms is greatest at around the same range of dimensions.

5.1.3.2 Tests on "Real-World" data

As regards experiments over real-word data we used the *Nasa* [8, 101] and *Colors* [101] SISAP benchmarks (see also Section 2.5). In each case, ten percent of the data is used as queries over remaining 90 percent of the set, at threshold values which return 0.01%, 0.1% and 1% of the datasets respectively. Table 5.3 summarises characteristics of these two datasets as well as reporting search thresholds.

Table 5.3: *SISAP datasets. Experimental threshold values that return the 0.01%, 0.1% and 1% of the datasets, respectively.*

Dataset	# elements	feature	dim	Used metric	Experimental Thresholds		
					$t_{0.01\%}$	$t_{0.1\%}$	$t_{1\%}$
Nasa	40,150	PCA-reduced Color Histograms	20	ℓ_2	0.120	0.285	0.530
Colors	112,682	Color Histograms	112	ℓ_2	0.052	0.083	0.131

There are many different contexts for metric search, and no mechanism is generally believed to be best for all purposes. For our experiment, we considered the Distal Spatial Approximation Tree (DiSAT) [65] which has been shown to perform better than a large range of other mechanisms. In [65], Chávez et

[5] See Sections 2.4.7.2, and 2.4.7.3 for the definitions of GHT, BST, and MBT.

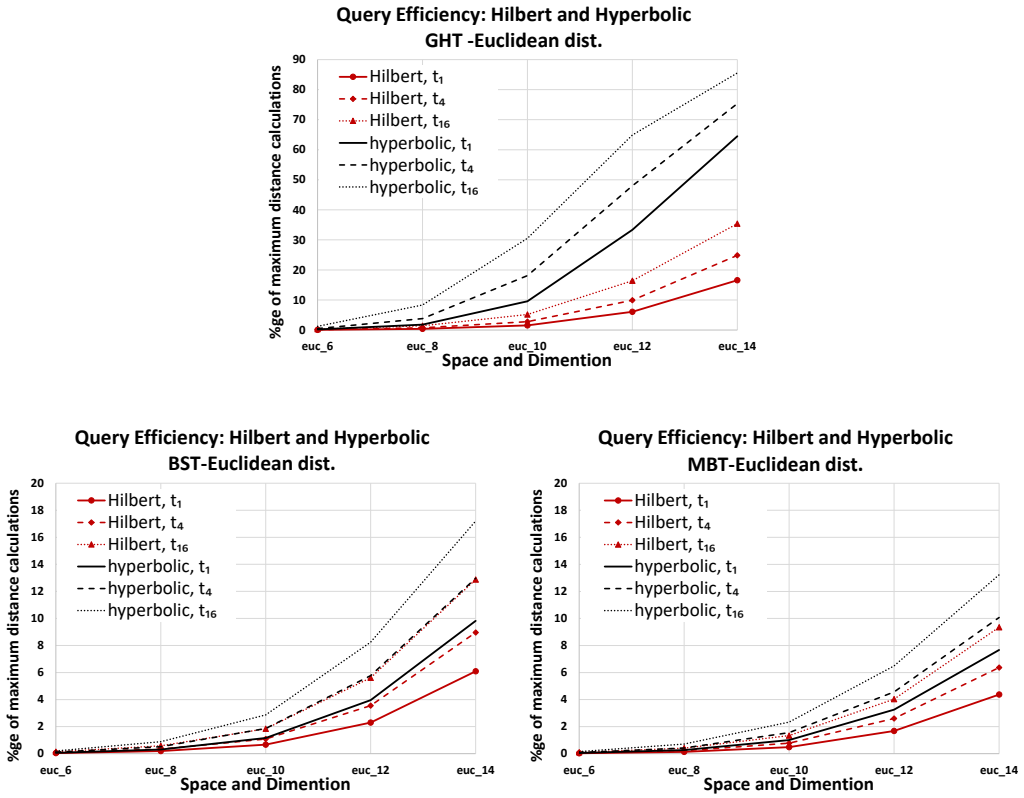


Figure 5.11: *Performance of three hyperplane-based metric index structures (GHT, BST and MBT) with the different exclusion strategies at various dimensionality and thresholds. The performance is measured as the mean number of distance calculations per query as percentage of the dataset size (1 million). In each graph, lines of the same pattern represent the same data, and the same index structures, only the query exclusion mechanism is different. Results are related to Euclidean metric spaces.*

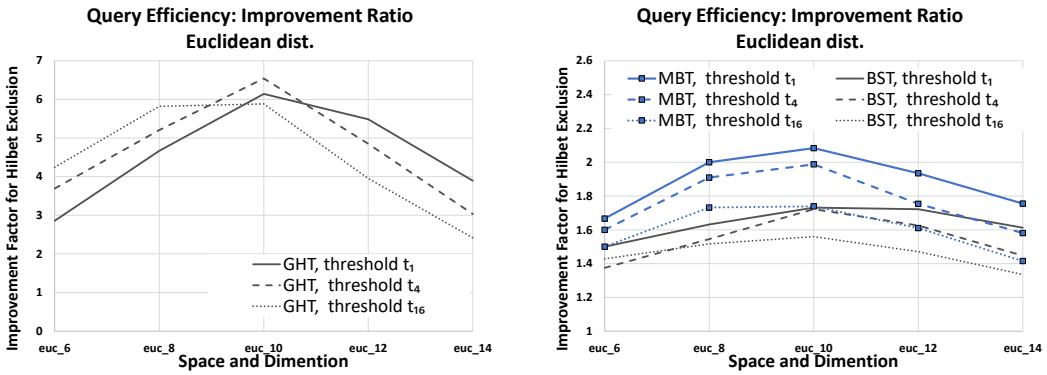


Figure 5.12: *Improvement ratio for Hilbert exclusion with generalized hyperplane tree (left) and bisector trees (right) on Euclidean space of various dimensions. The improvement is computed as the ratio of the number of distances calculated using Hilbert or hyperbolic at various dimensionality of Euclidean space. Jensen-Shannon and Triangular metrics give similar results.*

Table 5.4: *Cost for Generalized Hyperplane Tree (GHT), Bisector Tree (BST), and Monotonous Bisector Tree (MBT): Mean number of distance calculations per query as percentage of data size ($n = 10^6$). For each metric space, threshold t_k was experimentally derived as that which return around k results per million data.*

Space	GHT						BST						MBT					
	Hyperbolic			Hilbert			Hyperbolic			Hilbert			Hyperbolic			Hilbert		
	t_1	t_4	t_{16}	t_1	t_4	t_{16}	t_1	t_4	t_{16}	t_1	t_4	t_{16}	t_1	t_4	t_{16}	t_1	t_4	t_{16}
euc_6	0.20	0.48	1.23	0.07	0.13	0.29	0.06	0.11	0.20	0.04	0.08	0.14	0.05	0.08	0.15	0.03	0.05	0.10
euc_8	1.82	3.80	8.32	0.39	0.73	1.43	0.31	0.51	0.88	0.19	0.33	0.58	0.26	0.42	0.71	0.13	0.22	0.41
euc_10	9.58	18.12	30.53	1.56	2.77	5.19	1.16	1.86	2.87	0.67	1.08	1.84	1.00	1.55	2.33	0.48	0.78	1.34
euc_12	33.34	48.01	64.88	6.08	9.90	16.41	3.96	5.77	8.25	2.30	3.55	5.61	3.25	4.56	6.49	1.68	2.60	4.03
euc_14	64.50	75.36	85.55	16.57	24.86	35.41	9.82	12.97	17.20	6.09	8.96	12.88	7.67	10.07	13.23	4.37	6.37	9.35
jsd_6	0.19	0.54	1.59	0.06	0.14	0.30	0.05	0.11	0.21	0.04	0.08	0.15	0.04	0.08	0.16	0.03	0.05	0.11
jsd_8	1.84	5.01	10.96	0.36	0.86	1.80	0.29	0.58	1.05	0.18	0.37	0.71	0.24	0.46	0.84	0.13	0.27	0.52
jsd_10	13.16	25.47	42.15	2.34	4.63	8.93	1.58	2.72	4.51	0.96	1.75	3.14	1.45	2.40	3.84	0.73	1.37	2.47
jsd_12	37.63	58.97	75.67	7.60	14.81	24.49	4.69	7.73	11.63	2.88	5.31	8.74	4.20	6.74	9.91	2.24	4.18	7.01
jsd_14	72.97	85.51	92.36	24.16	37.05	50.98	13.17	18.78	25.23	9.27	14.62	21.43	11.35	16.05	21.68	7.35	11.77	17.67
tri_6	0.19	0.62	1.91	0.06	0.13	0.31	0.05	0.11	0.22	0.04	0.07	0.15	0.04	0.08	0.16	0.03	0.05	0.11
tri_8	2.93	8.59	19.46	0.50	1.19	2.63	0.41	0.81	1.48	0.24	0.49	0.96	0.33	0.64	1.15	0.17	0.36	0.72
tri_10	19.88	34.07	53.11	2.78	5.70	10.66	1.86	3.05	4.99	1.05	1.94	3.42	1.53	2.50	4.00	0.76	1.43	2.60
tri_12	50.75	68.91	82.09	10.66	20.09	32.30	6.14	9.71	14.26	3.80	6.95	11.26	5.28	8.25	11.92	2.92	5.45	9.11
tri_14	82.90	91.16	96.08	30.77	45.56	60.05	16.24	22.43	29.68	11.94	18.25	25.90	13.71	19.00	25.42	9.63	14.81	21.79

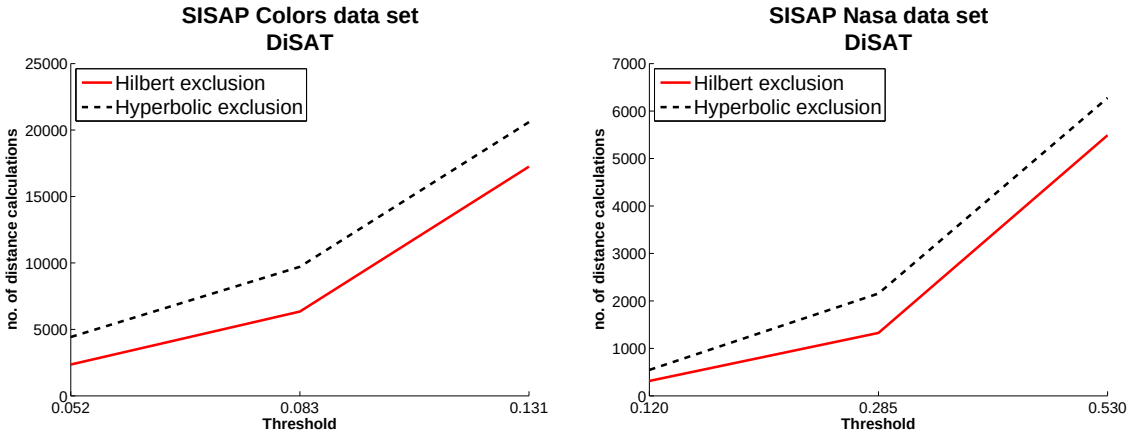


Figure 5.13: *Comparing hyperbolic and Hilbert exclusion Conditions for the DiSAT. The two graphs represent benchmark applications of the state-of-the-art DiSAT index over SISAP datasets, with significant improvements achieved through changing only the exclusion condition.*

al. assert: *"Our data structure has no parameters to tune-up and a small memory footprint. In addition, it can be constructed quickly. Our approach is among the most competitive, those outperforming DiSAT achieve this at the expense of larger memory usage or an impractical construction time."*

We therefore considered DiSAT as the state-of-the-art hyperplane-based indexing mechanism and we tested the effect of applying Hilbert exclusion against the hyperbolic exclusion with which DiSAT has been defined. Figure 5.13 shows the outcome of these experiments. Also in this case, we observe that Hilbert exclusion greatly improves the performance.

In [77] we have provided a full evaluation of the performance of the Hilbert exclusion within a variety of different hyperplane partition indexing structures, including a number of variants of DiSAT and bisector trees. One outcome of this evaluation is that the results obtained for different hyperplane partition trees vary a lot if using hyperbolic exclusion, while they are quite similar if using the Hilbert Exclusion. Moreover, the simple bisector tree with logarithmic arity (*i.e.* using a dynamic branching factor that does not exceed the natural logarithm of the data size) achieved the best performance over the SISAP benchmark datasets; to put this result in perspective, it required only around 40% of the number of distance calculations per query of the previous state of the art given in [65].

5.2 Metric Spaces and n-Point Property

In the previous section, we proposed the Hilbert exclusion, which shown promising improvement over the hyperbolic exclusion condition. However, we observed that our novel exclusion condition is applicable only when the metric space has the four-point property. The principal family of spaces that meet such property is the Hilbert-embeddable one. Actually, if a space is isometrically embeddable in a Hilbert space it also meets the more restrictive n-point property for any finite value of n. In the next subsections, we show the n-embeddability property for some important metrics as well as provide examples of non-embeddable spaces.

5.2.1 Isometrically n-Embeddable Spaces

This section uses various results gathered from mathematical literature to show that any metric space which is isometrically embeddable in a Hilbert space can be used with our exclusion condition.

The importance of Hilbert spaces is the generalisation of the notion of Euclidean space by extending the methods of vector algebra and calculus to spaces with any finite or infinite number of dimensions. A Hilbert space is an abstract vector space possessing the structure of an inner product that allows length

and angle to be measured, which gives certain geometric properties. These properties extend to abstract metric spaces which can be isometrically embedded in a Hilbert space. The key property of interest here is in n-point isometric embedding in ℓ_2^{n-1}.

Lemma 5.2.1 (Schoenberg's Theorem [229, 249])**.** *Let $(\mathcal{D}, \phi)$ a semi-metric space and ϕ such that, for all finite sets $\{c_i\}_{i\leq n}$ of real numbers and all finite sets $\{x_i\}_{i\leq n}$ of points in $\mathcal{D}$, the implication*

$$\sum_{i=1}^{n} c_i = 0 \Rightarrow \sum_{i,j=1}^{n} c_i c_j \phi(x_i, x_j) \leq 0 \tag{5.14}$$

holds (that is, ϕ is conditionally negative semidefinite function). Then $(\mathcal{D}, \sqrt{\phi})$ is a metric space which can be embedded isometrically as a subspace of a real Hilbert space.

Lemma 5.2.2 (Blumenthal Lemma 53.1 [51])**.** *A numerable semimetric space is isometrically embeddable in a Hilbert space if and only if it is isometrically n-embeddable in ℓ_2^{n-1} for every positive integer n.*

The main importance from our perspective is that, given a metric space $(\mathcal{D}, \sqrt{\phi})$, it is sufficient for ϕ to be a conditionally negative semidefinite function in order to have isometric embeddability into a Hilbert Space. Moreover the Hilbert embeddability guarantees the n-point property *for all* n.

5.2.1.1 $\mathbb{R}^k$ with Euclidean and Cosine Distances

From the above lemmata, directly follows that Euclidean spaces meet the n-point property since they are finite subspaces of a Hilbert space.

Proposition 5.2.1. *The Euclidean spaces of any dimension have the n-point property for any n, and can therefore use the Hilbert exclusion condition.*

A constructive proof can be provided showing that the function $\phi(\mathbf{v}, \mathbf{w}) = \sum_{i=1}^{n}(v_i - w_i)^2$ is conditionally negative definite which is straightforward to demonstrate.

Moreover, it follows that $\mathbb{R}^n$ with the Cosine Distance d_{Cos} (Equation 2.32) meets the n-point property as well.

Corollary. *$(\mathbb{R}^k, d_{Cos})$ has the n-point property for any n and k, and so the Hilbert exclusion Condition is valid over it.*

In facts, since $\|\mathbf{v} - \mathbf{w}\|^2 = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - 2\mathbf{v} \cdot \mathbf{w}$, the distance $d_{\text{Cos}}(\mathbf{v}, \mathbf{w})$ is equivalent to the Euclidean distance computed on the normalized vector $\mathbf{v}/\|\mathbf{v}\|$ and $\mathbf{w}/\|\mathbf{w}\|$:

$$d_{\text{Cos}}(\mathbf{v}, \mathbf{w}) = d_{\text{Cos}}\left(\frac{\mathbf{v}}{\|\mathbf{v}\|}, \frac{\mathbf{w}}{\|\mathbf{w}\|}\right) = \frac{1}{\sqrt{2}}\,\ell_2\left(\frac{\mathbf{v}}{\|\mathbf{v}\|}, \frac{\mathbf{w}}{\|\mathbf{w}\|}\right).$$

5.2.1.2 $\mathbb{R}^k$ with the Quadratic Form Distance

The class of spaces with the costly Quadratic Form Distance (QFD) (Section 2.4.1.1) has the n-point property:

Proposition 5.2.2. *$\mathbb{R}^k$ with the Quadratic Form Distance have the n-point property for any n and k, and can therefore use the Hilbert exclusion condition.*

This is a direct consequence of the Schoenberg's Theorem, since the (QFD) is the square root of the semi-metric

$$\phi(\mathbf{v}, \mathbf{w}) = (\mathbf{v} - \mathbf{w})^\top M (\mathbf{v} - \mathbf{w}), \tag{5.15}$$

where M is a semidefinite positive matrix (*i.e.* $\mathbf{z}^\top M\mathbf{z} \geq 0$, for any $\mathbf{z}$) . Clearly if $\{c_i\}_{i\leq n}$ is a sequence of real number that sum to zero, and $\{\mathbf{x}_i\}_{i\leq n}$ is a sequence of data points we have

$$\sum_{i,j=1}^{n} c_i c_j (\mathbf{x}_i - \mathbf{x}_j)^\top M(\mathbf{x}_i - \mathbf{x}_j) = \sum_{i,j=1}^{n} c_i c_j \mathbf{x}_i^\top M\mathbf{x}_i + \sum_{i,j=1}^{n} c_i c_j \mathbf{x}_j^\top M\mathbf{x}_j - 2\sum_{i,j=1}^{n} c_i c_j \mathbf{x}_i^\top M\mathbf{x}_j \tag{5.16}$$

$$= -2\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \mathbf{x}_i^\top M\mathbf{x}_j \tag{5.17}$$

$$= -2\left(\sum_{h=1}^{n} c_h \mathbf{x}_h\right)^\top M\left(\sum_{h=1}^{n} c_h \mathbf{x}_h\right) \leq 0. \tag{5.18}$$

5.2.1.3 Probability Distribution Space with Jensen-Shannon Distance

It is already known that the square root of the Jenson-Shannon divergence is a proper metric on the space of probability distributions [95, 104, 201] (see also Section 2.4.1.1)). Here we bring attention to the fact that it has the n-point property.

Lemma 5.2.3 (Topsøe [104])**.** *The space $(M_+^1(A), d_{JSD})$ is isometrically isomorphic to a subset in Hilbert Space.*

Topsøe uses Schoenberg's conjecture to prove this property by showing that Jenson-Shannon divergence is itself a negative semidefinite mapping with the semi-metric properties. Thus it follows

Proposition 5.2.3. *The space $(M_+^1(A), d_{JSD})$ is has the n-point property for any n, and can therefore use the Hilbert exclusion condition.*

5.2.1.4 Probability Distribution Space with Triangular Distance

To establish the generality of our results, we give one more example of a proper metric which is also Hilbert space embeddable and can, therefore, be searched using the Hilbert exclusion. In particular, we consider the Triangular Distance (Equation (2.34), Section 2.4.1.1). We report the mathematical proof in order to give an example of how to explicitly prove that a function is conditionally negative semidefinite.

Proposition 5.2.4. *The space $(M_+^1(A), d_\Delta)$ has the n-point property for any n, and can therefore use the Hilbert exclusion condition.*

Proof. For the Schoenberg's Theorem it is sufficient to prove that the Triangular Discrimination, defined as

$$\Delta(\mathbf{v}, \mathbf{w}) = \sum_i \frac{(v_i - w_i)^2}{v_i + w_i} \quad \forall\, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n, \sum_i v_i = \sum_i w_i = 1,$$

is a conditionally negative semidefinite function. Moreover, as Δ is a summation it is sufficient to prove that $f(x, y) = \dfrac{(x-y)^2}{x+y}$, where $x, y \in \mathbb{R}$, is conditionally negative semidefinite.

Recalling the definition of negative semidefinite (Equation 5.19) we require

$$\sum_{i,j} \frac{(x_i - x_j)^2}{x_i + x_j} c_i c_j \leq 0$$

for any finite set of real numbers $\{c_i\}_{i\leq m}$ such that $\sum_i c_i = 0$ and for any finite set $\{x_i\}_{i\leq n}$ of points in $\mathbb{R}$.

Observing that $(x_i - x_j)^2 = (x_i + x_j)^2 - 4x_i x_j$ we obtain

$$\sum_{i,j}^{m} c_i c_j \frac{(x_i - x_j)^2}{x_i + x_j} = \sum_{i,j}^{m} c_i c_j x_i + \sum_{i,j}^{m} c_i c_j x_j - 4\sum_{i,j}^{m} c_i c_j \frac{x_i x_j}{x_i + x_j} = -4\sum_{i,j}^{m} c_i c_j \frac{x_i x_j}{x_i + x_j}$$

as the first two terms sum to zero. Thus it is sufficient to prove that

$$\sum_{i,j}^{m} c_i c_j \frac{x_i x_j}{x_i + x_j} \geq 0$$

As the index i, j such $x_i = 0$ or $x_j = 0$ do not contribute to the summation, we can assume that all the x_i, x_j are positive.

$$\begin{aligned}
\sum_{i,j}^{m} c_i c_j \frac{x_i x_j}{x_i + x_j} &= \sum_{i,j}^{m} c_i c_j x_i x_j \int_0^\infty e^{-t(x_i+x_j)} dt \\
&= \int_0^\infty \sum_{i,j}^{m} c_i c_j x_i x_j e^{-t(x_i+x_j)} dt \\
&= \int_0^\infty \left(\sum_{i}^{m} c_i x_i e^{-t x_i} \right) \left(\sum_{j}^{m} c_j x_j e^{-t x_j} \right) dt \\
&= \int_0^\infty \left(\sum_{i}^{m} c_i x_i e^{-t x_i} \right)^2 dt \geq 0
\end{aligned}$$

This therefore gives us that the Triangular Distance $d_\Delta(v, w) = \sqrt{\Delta(v, w)}$ is a proper metric such that $(M_+^1(A), d_\Delta)$ is isometrically embeddable in Hilbert space. □

5.2.2 Supermetrics: Isometrically 4-Embeddable Spaces

Since our new exclusion condition requires just the four-point property, we are interested also in spaces that satisfy conditions less restrictive than the Hilbert-embeddability.

In [51] a weaker version of the Schoenberg's theorem is used to characterize any metric space which has the four-point property :

Lemma 5.2.4 ([51]). *A metric space $(\mathcal{D}, d)$ is isometrically 4-embeddable in ℓ_2^3 if and only if for all set $\{c_1, c_2, c_3, c_4\}$ of real numbers and all finite sets $\{x_1, x_2, x_3, x_4\}$ of points in $\mathcal{D}$, the implication*

$$\sum_{i=1}^{4} c_i = 0 \Rightarrow \sum_{i,j=1}^{4} c_i c_j d(x_i, x_j)^2 \leq 0 \tag{5.19}$$

holds.

In general, any inner product space meets this property:

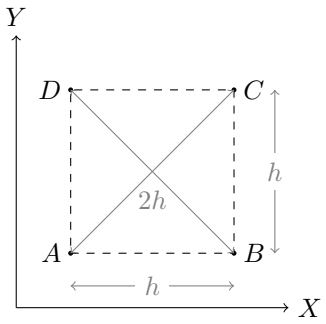
Lemma 5.2.5 (Scholtes Proposition 1.3 [230]). *Let $(\mathcal{V}, \|\cdot\|)$ be a normed vector space. Then the following statements are equivalent:*

- *$(\mathcal{V}, \|\cdot\|)$ is an inner product space,* i.e.*, there exists an inner product $<\cdot,\cdot>$ on $\mathcal{V}$ which induces the norm: $\forall\, \mathbf{x} \in \mathcal{V},\ \|\mathbf{x}\| = \sqrt{<\mathbf{x}, \mathbf{x}>}$*
- *there exists an inner product space $(\mathcal{W}, \|\cdot\|_\mathcal{W})$ such that $(\mathcal{V}, \|\cdot\|)$ is isometrically embeddable in $(\mathcal{W}, \|\cdot\|_\mathcal{W})$*
- *all subsets $\{\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{x}\} \subset \mathcal{V}$ are isometrically embeddable in ℓ_2^3.*

Interestingly, in [51], it is shown that

Lemma 5.2.6. *If $(\mathcal{D}, d)$ is a metric space then $(\mathcal{D}, d^\alpha)$, with $0 \leq \alpha \leq 1/2$, is isometrically 4-embeddable in ℓ_2^3.*

This implies, for example, that for any proper metric, a new metric with the four-point property can be formed by taking its square root, and thus used in a metric search structure with Hilbert exclusion. However, for practical purposes, this result is unlikely to be useful. In facts, raising a metric to a small power may cause an increase in its intrinsic dimensionality and thus a higher probability of no exclusion being made.

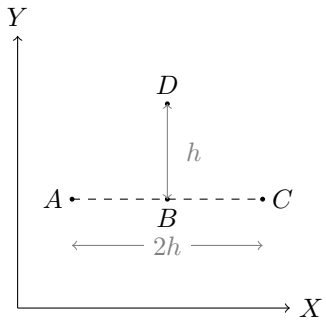


(a)

	A	B	C	D
A	0	h	$2h$	h
B		0	h	$2h$
C			0	h
D				0

(b)

Figure 5.14: *Showing that ℓ_1 does not have the four-point property. On the left, we plot four points in $(\mathbb{R}^2, \ell_1)$ that are not embeddable in $(\mathbb{R}^3, \ell_2)$. On the right, we show the ℓ_1 distances between the points. Any isometric embedding in ℓ_2^3 maps A, B, C to collinear points, meaning that the point D cannot be embedded whilst preserving the distances.*



(a)

	A	B	C	D
A	0	h	$2h$	h
B		0	h	h
C			0	h
D				0

(b)

Figure 5.15: *The ℓ_∞ distance also does not have the four-point property. On the left, we plot four points in $(\mathbb{R}^2, \ell_\infty)$ which are not embeddable in $(\mathbb{R}^3, \ell_2)$. On the right, we report the ℓ_∞ distances between the points. Once again, any isometric embedding must map A, B, C to collinear points in ℓ_2^3 making D impossible to embed.*

5.2.3 Non-Embeddable Spaces

To complete the picture, it is worth mentioning that not all metric spaces are four-embeddable in ℓ_2^3; it is therefore necessary to make a proper assessment of the space in question before using Hilbert exclusion.

Manhattan (ℓ_1), Chebyshev (ℓ_∞), Hamming and Edit distances are notable examples of metrics that do not meet the four-point property. In fact, it is easy to provide a counterexample for each of these metrics.

Let's first consider the Manhattan distance. Figure 5.14 shows, on the left, a four points in the 2D Cartesian plane and, on the right, the ℓ_1 distances between those points. Since $\ell_1(A, C) = \ell_1(A, B) + \ell_1(B, C)$, for any isometric embedding of A, B, C in ℓ_2^3 the three points are collinear. However, this is also true for A, D and C, and the distances are the same. Therefore, the four points A, B, C, D cannot be isometrically embedded in ℓ_2^3, as this would require that $\ell_1(B, D) = 0$.

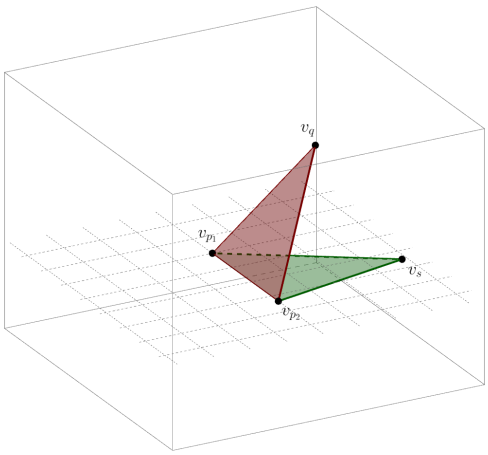
Similarly in Figure 5.15, $\ell_\infty(A, C) = \ell_\infty(A, B) + \ell_\infty(B, C)$ and so the points are collinear for any isometric embedding in ℓ_2^3; again, D shares the same distances ($\ell_\infty(A, C) = \ell_\infty(A, D) + \ell_\infty(D, C)$), and so a four-point embedding again cannot be achieved as this would require $\ell_\infty(B, D) = 0$.

To show that Hamming distance does not have the four-point property, it can simply be noted that the same distance table as that in Figure 5.14 is generated by the values $A = 00, B = 10, C = 11, D = 01$. In fact, this counterexample gives the same distances also for Edit distance, which therefore does not have the four-point property.

5.3 Tetrahedral projection onto a plane

As discussed previously, there exists a large class of metric space that meet the four-point property. This property has allowed us to define a novel exclusion mechanism which can be used to improve standard hyperplane partitioning. In this section, we continue our investigation on the four-point property. Specifically, we show how to use the isometric 4-embeddability in 3D Euclidean space to compute a lower-bound of the actual distance which allows us to define novel partitioning approaches that are possible only on supermetric spaces.

We start from the observation that in the same manner with the triangle inequality allows us to draw any three metric objects as the vertices of a triangle in a 2D Euclidean space (Section 5.1.2), the four-point property allows drawing any four objects as the vertices of a tetrahedron in 3D. However, the practical case is that we know only five out of the sixth edge lengths of the tetrahedron because only five of the six possible distances have been measured in the original metric space. This corresponds to the situation of an indexing structure based on two reference objects, p_1 and p_2, which were used to partition a dataset $\mathcal{S} \subset \mathcal{D}$ according to relative distances from these objects. The third object s represents an arbitrary element of $\mathcal{S}$ which has been stored, and the fourth and final object q represents a query over the data. For all possible data objects, we wish to identify those which may be within a threshold distance t of q, based on some partition of the space constructed before q was available. The three distance $d(p_1, p_2), d(s, p_1)$ and $d(s, p_2)$ are calculated during the index build process and used to guide the structuring of the data. At query time, for a query q, the two distances $d(q, p_1)$ and $d(q, p_2)$ are calculated and may be used to make some deduction relating to this structure. This situation gives knowledge of two adjacent faces of the tetrahedron which can be formed in ℓ_2^3, whose common base is the edge $\overline{\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}}$, where for each object a the notation $\boldsymbol{v}_a$ is used to denote the corresponding point in ℓ_2^3:



Neither the edge $\overline{\boldsymbol{v}_q\boldsymbol{v}_s}$ nor the angle between these two triangles are known without explicitly computing the distance $d(s, q)$ in the original space. Since the triangle inequality gives us upper and lower bounds on this distance (Double-Pivot distance constrain, Section 2.4.6.4), we question whether the tetrahedral embedding can provide distance bounds as well. It is immediate to show that no matter what the angle between the two triangles is, if we rotate the triangle $\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}\boldsymbol{v}_s$ around the line $\overline{\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}}$ until it is coincident with the plane where $\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}\boldsymbol{v}_q$ lies we obtain two possible projections, each providing us a distance bound. Figure 5.16 shows the described rotation procedure. The two possible orientations give the upper and lower bounds, corresponding to the distances between $\boldsymbol{v}_s$ and the two apexes $\boldsymbol{v}_q^-$ and $\boldsymbol{v}_q^+$ of the two possible planar tetrahedra. Therefore if $\ell_2(\boldsymbol{v}_q^+, \boldsymbol{v}_s) > t$ then also $\ell_2(\boldsymbol{v}_q, \boldsymbol{v}_s) = d(s, q) > t$, which means that s is not a solution. Similarly if $\ell_2(\boldsymbol{v}_q^-, \boldsymbol{v}_s) \leq t$ than $d(s, q) \leq t$.

The coordinates of the vertices in the planar plane can be easily computed as done in Section 5.1.2 (see also Figure 5.6). Many such coplanar triangles can be depicted, representing many points of a space, in a single scatter plot. For example, Figure 5.17 shows a set of 500 points, drawn from randomly

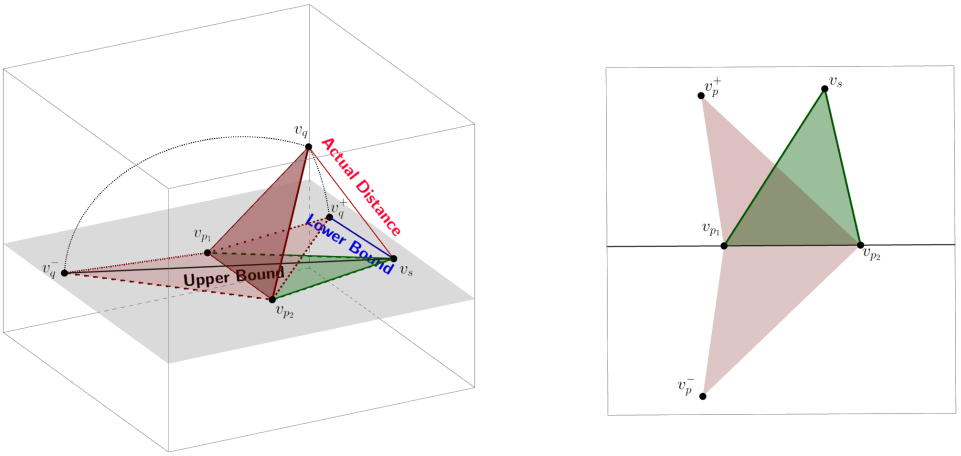


Figure 5.16: *Projection of two adjacent triangles onto the same plane by rotating one of the two around the common basis.*

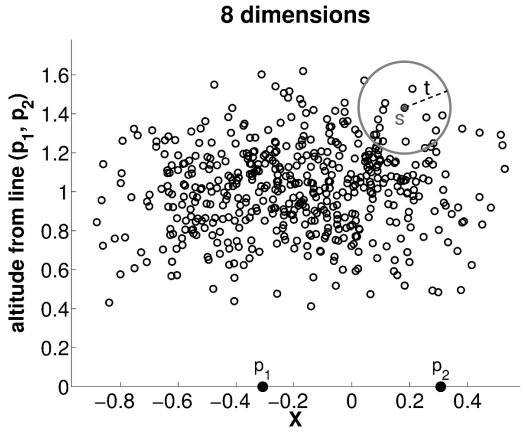


Figure 5.17: *Scatter diagram for 8-dimensional Euclidean Space. The distance δ between two selected reference points p_1 and p_2 is measured, and an embedding function is chosen which maps these to $(0, -\delta/2)$ and $(0, \delta/2)$ respectively. Other points s_i in the space are then plotted to preserve the distances $d(s_i, p_1)$ and $d(s_i, p_2)$. For metric spaces with the four-point property, the ℓ_2 distance between the corresponding points in this diagram is a lower bound on $d(s_i, s_j)$ in the original space. Hence, any point within t of a point s in the original space cannot lie outside the circle of radius t centered around s in the scatter plot.*

generated 8-dimensional Euclidean space, and plotted with respect to their distances from two fixed reference points p_1 and p_2. The distance between the reference points is measured, and the reference points are plotted on the X-axis symmetrically either side of the origin. For each point in the rest of the set, the distances $d(s, p_1)$ and $d(s, p_2)$ are calculated and used to plot the unique corresponding point in a triangle above the X-axis, according to these edge lengths.

It is important to be aware, in this and the following figures, of the importance of the four-point property. In Section 5.1.2, we saw that the same diagram can be plotted for a simple metric space (thanks to triangle inequality), but in this case no spatial relationship is implied between any two points plotted in the 2D plane: no matter how close two points are in the plot, there is no implication for the distance between them in the original space. However, if the diagram is plotted for a supermetric space, then the distance between any two points on the plane is a lower bound on their distance in the original space; two points that are further than t on the plot cannot be within t of each other in the original space. This observation leads to an arbitrarily large number of ways of partitioning the space and allowing these partitions to be excluded on the basis of the query position in the 2D scatter plot and has many potential uses in metric indexing.

5.3.1 Indexes Based on Planar Projection with the Four-Point Property

The planar projection of a subset of a supermetric space into a 2D Euclidean space allow exploiting novel partitioning strategies that can be used to build an index of the data object.

Let us consider the case of a single level of partitioning of a space $\mathcal{S} \in \mathcal{D}$. Two reference objects p_1 and p_2 are selected in the original space and used to project any object $s \in S$ in a 2D plane, that is $s \mapsto \boldsymbol{v}_s$ where the coordinate of $\boldsymbol{v}_s \in \mathbb{R}^2$ are determined only by the distances $d(s, p_1)$, $d(s, p_2)$, and $d(p_1, p_2)$. Now any rule based on the geometry of this Euclidean plane can be used to partition the data in two or more subsets $\mathcal{S}_1, \ldots, \mathcal{S}_n$. For example, we can use a Euclidean hyperplane to split the data into two subsets, or a Voronoi-like partitioning of the 2D space. We assume that the rule used to partition the 2D vector data is not pathological in the sense that objects that fall in a partition are geometrically separated from the objects of the other partitions by a line, a curve or any well-defined geometric boundary. At query time a query object q is projected in the same 2D plane used in the index phase (*i.e.* using the same couple of reference points p_1 and p_2). The query will fall in one of the partitions of the data, suppose $\boldsymbol{v}_q \in \mathcal{S}_i$. Since we are now working on 2D Euclidean space it will be likely easy to compute the Euclidean distance between the query and the boundaries of the neighbouring partitions. So if the Euclidean distance between $\boldsymbol{v}_q$ and the boundary in 2D of the partition $\mathcal{S}_j$ is bigger than the query threshold t then the lower bounding property ensures us that $d(q, x) > t$ for all $x \in \mathcal{S}_j$, and therefore the partition $\mathcal{S}_j$ can be excluded from the search.

The described general approached can be used recursively to construct an index. In theory, each level of the data partitioning can use a different set of reference points and a different 2D partitioning rule. Since, as it will be shown, different spaces give quite different distributions of points within the plane, build-time partitions can be chosen according to this distribution, rather than as a fixed attribute of an index mechanism.

For the sake of simplicity, we consider the case in which a set of data is divided into precisely two partitions. The simplest such mechanism to consider is the application of this concept to normal hyperplane partitioning with Hilbert exclusion. Let p_1 and p_2 two pivots used to project the data in a 2D plane. We obtain a scatter plot similar to that used in Figure 5.17. Splitting the 2D data over the Y-axis corresponds to divide the original data set according to which of the points p_1 and p_2 is the closer. At query time, if the corresponding plot position for the query is further than t from the Y axis, no solutions can exist in the subset closer to the opposing reference point (which corresponds to the Hilbert Exclusion). Figure 5.18 shows this situation in a scatter diagram built from an 8-dimensional Euclidean space. The points that are drawn in solid, either side of the Y-axis, are distant more than the query threshold from the Y-axis; therefore, if they were query points, the opposing semi-space would not require being searched.

For randomly generated, evenly distributed points there seems to be little to choose. However, it is often the case that “real world" datasets do not show the same properties as uniform sets; in particular, they tend to be much less evenly distributed, with significant numbers of clusters and outliers. These

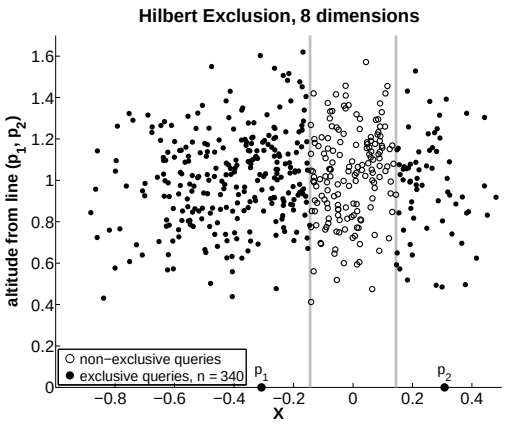


Figure 5.18: *Scatter diagram for 8-dimensional Euclidean Space. The data is divided into two subsets according to which side of Y-axis they lie. The solidly-coloured points are points that, were they queries, would allow the semispace on the opposing side to be excluded from the search since that semispace cannot contain a solution.*

factors can significantly affect the performance of indexing mechanisms. Here we underline the fact that we can choose arbitrary geometric partitions of the 2D plane to structure the original data.

Figures 5.19, 5.20, and 5.21 show a sample taken from the SISAP *Colors* dataset with Euclidean distance applied, showing six different partition of the 2D plane each dividing the space in exact halves, thus allowing a balanced search structure, as follows:

- *Figure 5.19a.*The data is partitioning according to a vertical line (parallel to the Y-axis) passing through the median of the point cloud, which corresponds to partition the data according to its X-coordinate, with reference to the median X-coordinate. A query whose projection has an X-coordinate greater than t from the this does not require to search the opposing half.

- *Figure 5.19b.* The data is partitioning according to a horizontal line (parallel to the X-axis) passing through the median point in the scatter plot, which corresponds to partition the data according to its Y-coordinate. A query whose projection has a Y-coordinate greater than t from the median Y-coordinate does not require to search the opposing half.

- *Figure 5.20a.* The data is partitioned according to the median distance d_m from the mid-point $\boldsymbol{v}_c$ of the point cloud. If the projected query $\boldsymbol{v}_q$ is such that

$$(\ell_2(\boldsymbol{v}_q, \boldsymbol{v}_c) < d_m - t) \vee (\ell_2(\boldsymbol{v}_q, \boldsymbol{v}_c) > d_m + t)$$

  then half of the space can be excluded.

- *Figure 5.20b.* The data is partitioned according to the median distance d_m from the top-left point of the point cloud. Exclusion conditions are defined similarly to the previous case.

- *Figure 5.21a.* The data is split using the best-fit line through the point cloud, that is computed using a least square minimization approach. The exclusion condition is defined on the basis of the distance of the projected query to this line.

- *Figure 5.21a.* Similarly to the previous case the data is partitioned by using a line orthogonal to the best-fit line through the point cloud.

The query threshold illustrated is 0.052 corresponding to a query returning 0.001% of the data.

In Figure 5.19 it can be seen that, for this sample data, partitioning the plane according to the Y-coordinate is the more effective then partitioning plane according to the X-coordinate. The disadvantage with this is that a little more calculation is required to plot the height of the point, rather than its offset

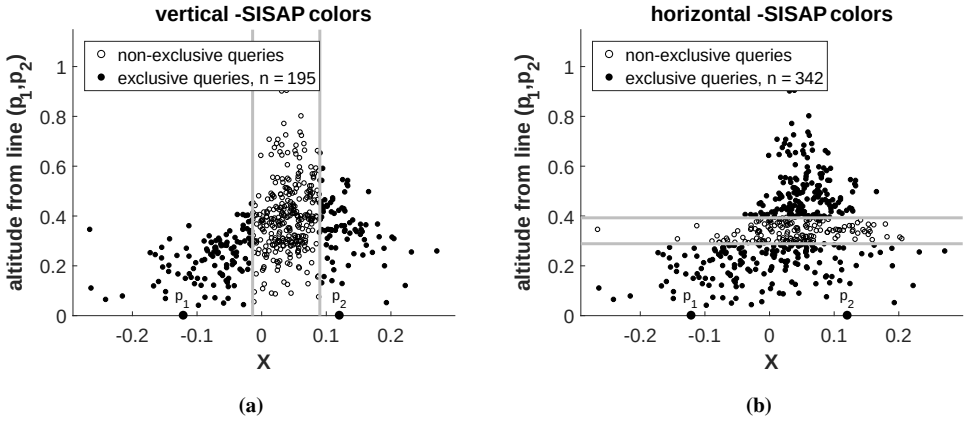


Figure 5.19: *Scatter diagrams dividing the plane equally in X and Y dimension, either can be used for partitioning a hyperplane tree structure; in this case, the horizontal partition would be more effective.*

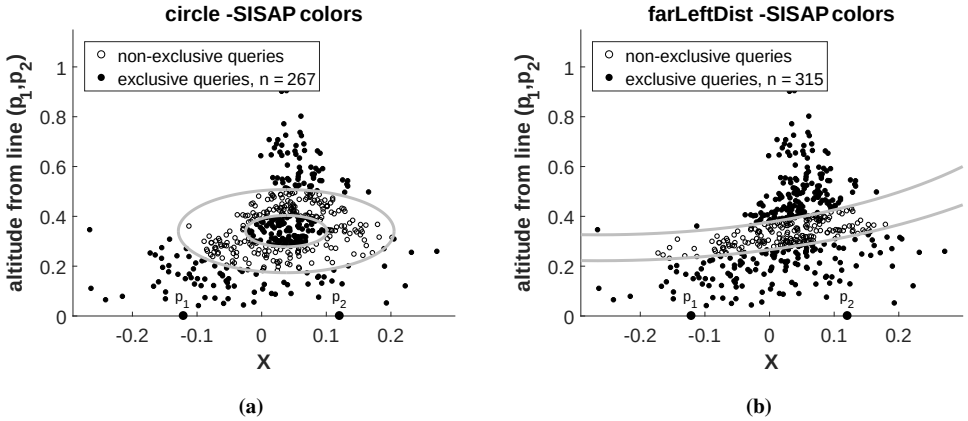


Figure 5.20: *Two more binary partitions, based now on median distance from arbitrary points in the plane (centre and top-left respectively).*

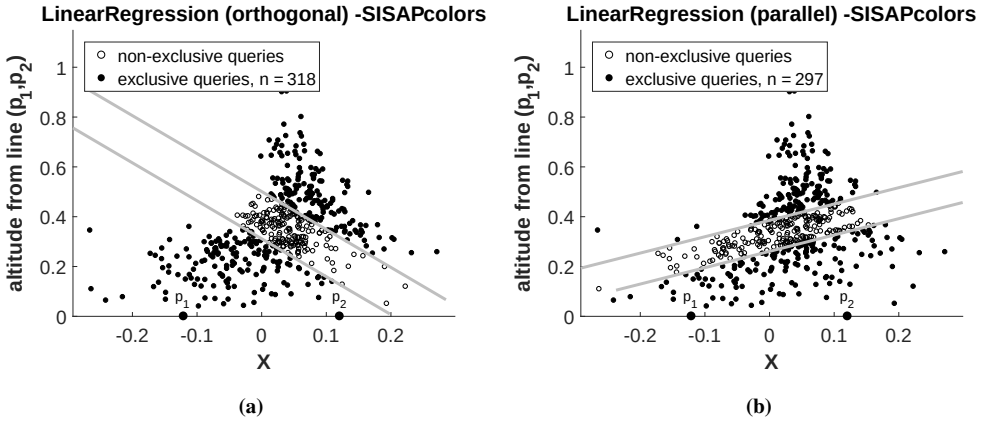


Figure 5.21: *Hyperplane partitioning based on a hyperplane orthogonal (left) or parallel (right) to the best-fit line through data.*

from the Y-axis; however, this is a minor effect when significantly more distance calculations can be avoided.

Partitions in Figure 5.20 are considered just to make the point that any partition of the plane can be used for this purpose. We have not found a compelling use for either partition, however, this would depend on the nature of an individual non-uniform dataset.

Figures 5.21 illustrate more technical analyses of the point cloud, using the Linear Regression (LR), that is discussed in more detail in the next section. This technique can be used along one of two axes in a two-dimensional space as illustrated in the Figure.

5.3.2 The Linear Regression Tree

In this section, we aim to stress the fact that the planar projection of a supermetric space can be used to define innovative partitioning and indexing schemas. To this scope, we present a completely novel indexing structure which is only possible to use in a supermetric space, where the hyperplane partition and consequent exclusion mechanism are dynamically chosen according to the distribution of data within each individual node of the tree.

Figure 5.21 shown a scatter plot resulting from an arbitrary choice of reference points for the SISAP *colors* dataset. Although the pattern is not atypical, observation shows that the individual distribution shape is significantly affected by the choice of reference points and, more subtly, by the subset of data points that is to be stored at a given tree node. The partitions shown within the figure are based on the best-fit straight line which can be plotted through the points in two dimensions. This is parallel to the lines drawn in the right-hand figure. As this is calculated using the least-mean-squares algorithm, it is reasonable to assume that the perpendicular partition, shown in the left-hand diagram, will in general improve the spread of the data points and thus form a better partition for indexing.

To test this strategy, we define the Linear Regression Tree (LRT), which is a binary monotone tree built recursively over a dataset S as follows. We select two reference points p_1, p_2 at each node. Each child node of the tree shares one reference points with its parents, as done in Monotonous Bisector Tree (MBT). We used the tetrahedral projection based on p_1 and p_2 to embed the data points onto a 2D plane, and we compute the best-fit line l through the projected points (or a subset of them) using a least squares minimization. Then, we rotate the 2D data points around the X-intercept of the line l, so that the new X-axis coincides with the line l, and we split the data at a certain value of the X coordinate of the rotated space.

Algorithm 1 and 3 give the simplest algorithms for constructing, and querying a balanced version of the LRT, where the data split is at the median X coordinate of the rotated space.

We compute the best fitting line l through the points $\{(x_i, y_i)\}_{i=1}^m$ as the line $y = cx + h$ that best fits the sample in the sense that the sum of the squared errors between the y_i and the line values $cx_i + h$ is minimized. The fitting line is easily computed as $y - \bar{y} = c(x - \bar{x})$ where $\bar{x} = \sum_{i=1}^m x_i/m$, $\bar{y} = \sum_{i=1}^m y_i/m$, and

$$c = \frac{\sum_{i=1}^m (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^m (x_i - \bar{x})^2}. \tag{5.20}$$

Then, we rotate the data points by angle $\theta = \arctan(c)$ around the X-intercept $(h, 0)$, where $h = \bar{x} - \bar{y}/c$:

$$r_x = (x - h)\cos(\theta) - y\sin(\theta) \tag{5.21}$$

$$r_y = (x - h)\sin(\theta) + y\cos(\theta). \tag{5.22}$$

Each node of the tree can also store minimum and maximum distances to either reference points to allow range-pivot exclusion as done in the MBT.

Experimental evaluation of the resulting search index was performed on SISAP benchmarks using exactly the same settings as described in Section 5.1.3.2. Figures 5.22 and 5.23 give results for the SISAP *Colors* and *Nasa* datasets respectively. For each dataset, six different indexing structures were tested. A balanced monotone bisector tree (MBT-balanced), an unbalanced monotone bisector tree (MBT-balanced), a balanced Linear Regression Tree (LRT-balanced), and a naive unbalanced version of the LRT where we split the data using the value $\lambda = h + d(p_1, p_2)/2$, which corresponds to modify only

Algorithm 1: CreateNode (LRT balanced)

Input : $A \subset S$, $p_1 \in S$

Output: Node: $N = \left\langle p_1, p_2,\ \lambda,\ \theta,\ h,\ N_{\text{left}}, N_{\text{right}}, cr^{+}_{\text{left}}, cr^{+}_{\text{right}}, cr^{-}_{\text{left}}, cr^{-}_{\text{right}} \right\rangle$ where
$\{p_1, p_2\} \subset U$, $\lambda \in \mathbb{R}$, $\theta \in [0, 2\pi)$, $h \in \mathbb{R}$, $\{N_{\text{left}}, N_{\text{right}}\} \subset$ Node, $cr^{\pm}_{*} \geq 0$

Select p_2 from A;
if $|A| > 2$ **then**
  $A \leftarrow A \setminus \{p_1, p_2\}$;
  $\widetilde{A} \leftarrow$2Dproject(A, p_1, p_2);
  $(\theta, h) \leftarrow$GetRotationAngle($\widetilde{A}$); // Calculate the rotation angle θ, and the X-intercepts $(h, 0)$, that minimize the squared errors of the Y-coordinates following the rotation transformation. If $\widetilde{A}$ is too large the linear regression is performed on a subset of it
  $RotatedPoints \leftarrow \emptyset$;
  foreach $\widetilde{s_j}$ *in* $\widetilde{A}_i$ **do**
    $r_j \leftarrow$ Rotate($\widetilde{s_j}, \theta, h$) ; // $r_j = (r_j.x,\ r_j.y) \in \mathbb{R}^2$
    $RotatedPoints \leftarrow RotatedPoints \cup \{r_j\}$
  end
  $\lambda \leftarrow$ median$\{r_j.x |\ r_j \in RotatedPoints\}$; // Find the median value of the X-coordinate of the rotated points, e.g bt using QuickMedianSort algorithm
  $A_{\text{left}} \leftarrow \{s_j \in A |\ r_j.x, \leq \lambda\}$;
  $A_{\text{right}} \leftarrow \{s_j \in A |\ r_j.x, \geq \lambda\}$;
  $cr^{+}_{\text{left}} \leftarrow \max_{s \in A_{\text{left}}} d(s, p_1)$;
  $cr^{-}_{\text{left}} \leftarrow \min_{s \in A_{\text{left}}} d(s, p_1)$;
  $cr^{+}_{\text{right}} \leftarrow \max_{s \in A_{\text{right}}} d(s, p_1)$;
  $cr^{-}_{\text{right}} \leftarrow \min_{s \in A_{\text{right}}} d(s, p_1)$;
  $N_{\text{left}} \leftarrow$ CreateNode(A_{left}, p_1);
  $N_{\text{right}} \leftarrow$ CreateNode(A_{right}, p_2);
  $N \leftarrow \left\langle p_1, p_2,\ \lambda,\ \theta,\ h,\ N_{\text{left}}, N_{\text{right}}, cr^{+}_{\text{left}}, cr^{+}_{\text{right}}, cr^{-}_{\text{left}}, cr^{-}_{\text{right}} \right\rangle$;
end

Algorithm 2: 2Dproject (2D projection of A based on $p_1,\ p_2$)

Input : $A \subset S$, $p_1, p_2 \in S$

Output: Set $\widetilde{A} \subset \mathbb{R}^2$

$\widetilde{A} \leftarrow \emptyset$;
$\delta \leftarrow d(p_1, p_2)$;
foreach s_j *in* A **do**
  /* Calculate the 2D embedded point $\widetilde{s_j}$ as the apex of the triangle defined by baseline $(0, -d(p_1, p_2)/2) - (0, d(p_1, p_2)/2)$, with left side length $d(s_j, p_1)$ and right side $d(s_j, p_2)$ */
  $\delta_1 \leftarrow d(s_j, p_1)$;
  $\delta_2 \leftarrow d(s_j, p_2)$;
  $\widetilde{s_j}.x \leftarrow (\delta_1^2 - \delta_2^2)/2\delta$;
  $\widetilde{s_j}.y \leftarrow$ sqrt $\left(\frac{\delta_1^2 + \delta_2^2}{2} - s_j.x^2 - \delta^2/4\right)$;
  $\widetilde{A} \leftarrow \widetilde{A} \cup \{\widetilde{s_j}\}$;
end

Algorithm 3: Query

Input : $q \in U,\ t \in \mathbb{R},\ N = \left\langle p_1, p_2,\ \lambda,\ \theta,\ h,\ N_{\text{left}}, N_{\text{right}}, cr^{+}_{\text{left}}, cr^{+}_{\text{right}}, cr^{-}_{\text{left}}, cr^{-}_{\text{right}} \right\rangle \in$ Node

Output: Result set $R = \{s \in S |\ d(s,q) \leq t\}$

$R \leftarrow \emptyset$;
if $d(q,\ p_1) \leq t$ **then**
  $R \leftarrow R \cup \{p_1\}$;
end
if $d(q,\ p_2) \leq t$ **then**
  $R \leftarrow R \cup \{p_2\}$;
end
if $\left(d(q,\ p_1) + t < cr^{-}_{left}\right) \vee \left(d(q,\ p_1) - t > cr^{+}_{left}\right)$ **then**
  $R \leftarrow R \cup Query(q, N_{\text{right}})$;
else
  if $\left(d(q,\ p_2) + t < cr^{-}_{right}\right) \vee \left(d(q,\ p_2) - t > cr^{+}_{right}\right)$ **then**
    $R \leftarrow R \cup Query(q, N_{\text{left}})$;
  else
    $\widetilde{q} \leftarrow$ 2Dproject$(\{q\}, p_1, p_2)$;
    $r_q =$ Rotate$(\widetilde{q},\ \theta,\ h)$;
    if $r_q.x < \lambda - t$ **then**
      $R \leftarrow R \cup Query(q, N_{\text{left}})$;
    else
      if $r_q.x > \lambda + t$ **then**
        $R \leftarrow R \cup Query(q, N_{\text{right}})$;
      else
        $R \leftarrow R \cup Query(q, N_{\text{left}})$;
        $R \leftarrow R \cup Query(q, N_{\text{right}})$;
      end
    end
  end
end

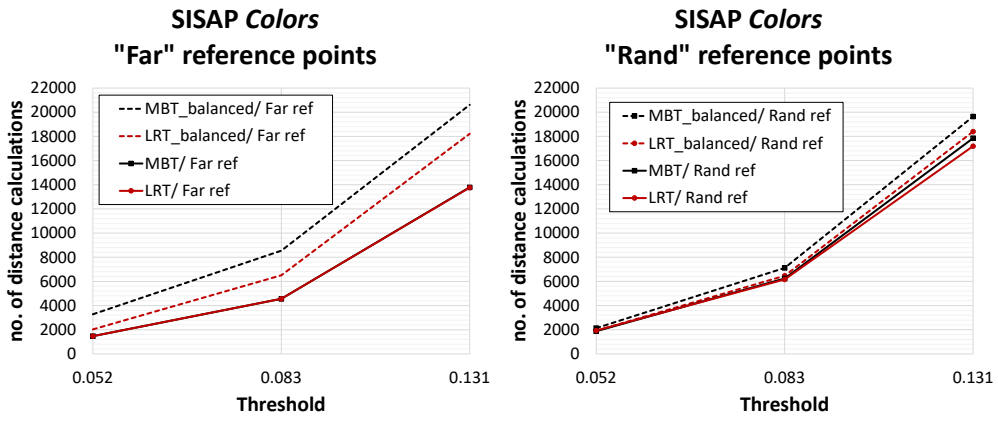


Figure 5.22: *SISAP Colors- Number of distance calculations for benchmark search thresholds using Monotonous Bisector Tree (MBT) and Linear Regression Tree (LRT) with two different reference point selection strategies, namely "Far" (on the left) and "Rand" (on the right). In the "Far" case the unbalanced MBT and Linear Regression Tree (LRT) have practically the same performance.*

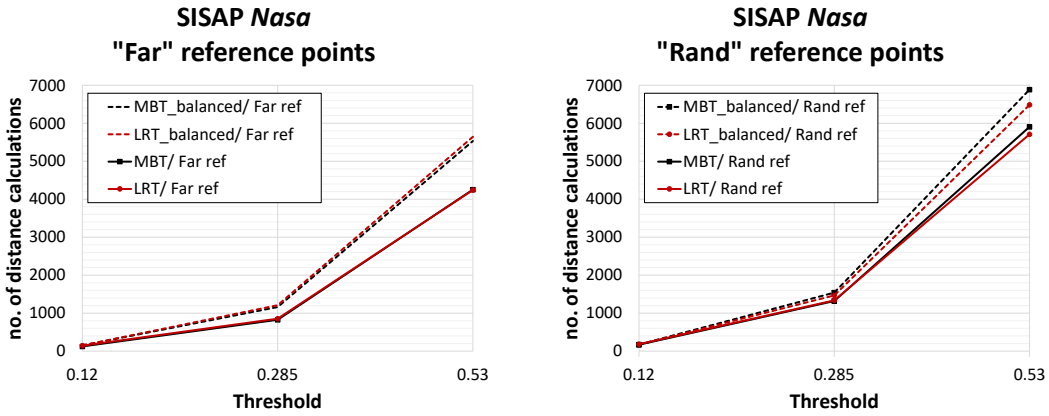


Figure 5.23: *SISAP Nasa- Number of distance calculations for benchmark search thresholds using Monotonous Bisector Tree (MBT) and Linear Regression Tree (LRT) with two different reference point selection strategies, namely "Far" (on the left) and "Rand" (on the right). In the "Far" case the MBT and LRT show quite the same performance.*

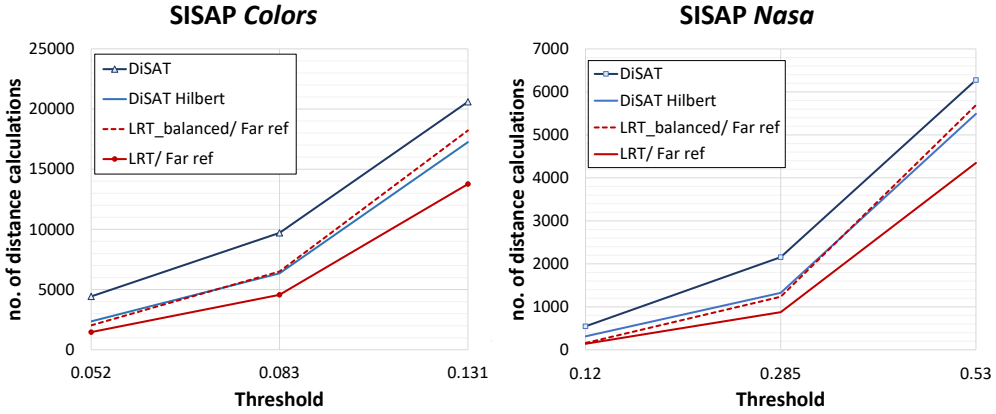


Figure 5.24: *SISAP Datasets - Number of distance calculations for benchmark search thresholds using Linear Regression Tree (LRT) and Distal Spatial Approximation Tree (DiSAT).*

the line 11 in the Algorithm 1. Each index was tested with two different reference point selection strategies. These are “Rand” and “Far”: at each node one reference point is handed down from an ancestor, and the second pivot is selected either randomly (“Rand" approach) or selecting the object at furthest distance from the first pivot within the data subset used to construct that node (“Far" approach). To a fair comparison, the MBT trees were tested using the Hilbert exclusion condition instead of the original hyperbolic exclusion.

It can be seen that among the balanced trees, the Linear Regression Tree always achieve better results than the Monotone Bisector Tree. The unbalanced versions have very similar performance and both outperform the balanced trees. However, it is worth noting that balanced structures are often slower than unbalanced ones for relatively small datasets, but they become rapidly more desirable as the size of the data increases, and again more so if it is too large to fit in main memory and requires to be stored in backing store pages.

Although the performance of LRT on the SISAP datasets does not differ much from that of MBT we believe that this data structure is an interesting example of how the distribution of the data can be iteratively exploited by means of the four-point property. We observe that the unbalanced LRT save more than the 30% of the number of the distance calculations per query of the previous state-of-the-art results given in [65] using DiSAT (see figure 5.24). Interestingly, our improved version of DiSAT that uses

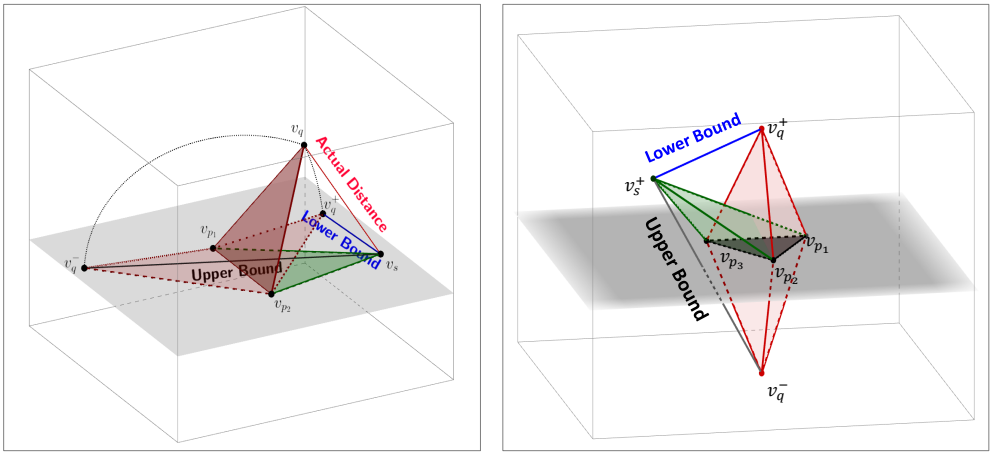


Figure 5.25: *Examples of n-simplex embedding. The left-hand figure shows the case $n = 2$, where the points (2 pivots and 2 data object) are first isometrically embedded in 3D Euclidean space and then projected in a 2D plane. The right-hand figure shows the case $n = 3$, where the points (3 pivots and 2 data objects) are first isometrically embedded in 4D Euclidean space and then projected in a 3D subspace (only the projected tetrahedrons are depicted in this case).*

Hilbert exclusion shows similar results to that of balanced LRT.

We finally observe that [76] reports experiments on large scale using 80-dimensional vector descriptors extracted from the CoPhIR [54] image dataset. In these experiments increasingly large subsets of the data, ranging from one million to sixteen million images, was queried to test the scalability of various search mechanisms, including unbalanced BST, log-sized monotone hyperplane tree, balanced VPT, and our balanced LRT that achieved the best overall performance. [6]

5.4 n-Simplex Projection

This section generalizes the tetrahedral/planar embedding introduced in the Section 5.3 in more dimensions. The motivation is that most of the supermetric spaces also satisfy the n-point property, as we observed in Section 5.2.1. This is the case of Hilbert-embeddable spaces, notable including those governed by Euclidean, Jenson-Shannon, Triangular and Quadratic Form distances.

The core observation here is that the n-point property guarantees that for any $(n+1)$ objects sampled from the original space, there exists an n-dimensional simplex in Euclidean space whose edge length correspond to the actual distance between the objects. A simplex is a generalisation of a triangle or a tetrahedron in arbitrary dimensions. In one dimension, the simplex is a line segment; in two it is the convex hull of a triangle, while in three it is the convex hull of a tetrahedron. In general, the n-simplex of vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n+1}$ equals the union of all the line segments joining $\boldsymbol{v}_{n+1}$ to points of the $(n-1)$-simplex of vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$.

In next sub-sections, we show how to construct these simplexes for metric spaces with the n-point property and how they can be used to derive arbitrarily tight upper and lower bounds on distances within the original space. We also show how to use this novel approach for similarity search and dimensionality reduction. Now we give an informal outline of our approach starting from the previously introduced tetrahedral/planar projection.

The left-hand diagram in Figure 5.25 repeat the earlier Figure 5.16, and shows the basic idea of the tetrahedral/planar projection: given two pivot p_1 and p_2 and two objects q and s of a supermetric space,

[6] The paper [76] presents several experiments that investigate extending the use of the supermetric property to larger and higher-dimensional data sets. The achieved results are very promising; however, we do not include them in this thesis as that set of experiments was performed principally by Prof. Richard Connor and our contribution was minimal.

it is always possible to *isometrically* embed them in 3D Euclidean space (ℓ_2^3); the projected points, indicated with $\boldsymbol{v}_{p_1}$, $\boldsymbol{v}_{p_2}$, $\boldsymbol{v}_s$, and $\boldsymbol{v}_q$, respectively, are the vertices of a tetrahedron (3-simplex). However, in a realistic case we do not know the distance $d(q,s)$ without explicitly compute it, and so we do not know the length of one edge of the tetrahedron. We showed that it is possible to use the knowledge of all the other edge length to provide upper and lower bound on the distance $d(q,s)$. Those bounds are obtained by rotating the triangle (2-simplex) of vertices $\boldsymbol{v}_{p_1}, \boldsymbol{v}_{p_2}, \boldsymbol{v}_q$ around the edge $\overline{\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}}$ (1-simplex) until it is coplanar with the other triangle. The two possible orientations give the upper and lower bounds, corresponding to the distances between $\boldsymbol{v}_s$ and the two apexes $\boldsymbol{v}_{q^-}$ and $\boldsymbol{v}_{q^+}$ of the two possible planar tetrahedra. The right-hand diagram in Figure 5.25 shows the same situation but starting from space that has the 5-point property. In this case we consider three pivots, p_1, p_2 and p_3, as well as two objects s and q. Also in this case we assume to know all the inter-object distances except for $d(q,s)$. As an outcome of the 5-point property, we can isometrically embed all the five points in 4D Euclidean space (ℓ_2^4). In this case, we know all the edge lengths of the two tetrahedra in 4D of vertices $\{\boldsymbol{v}_{p_1}, \boldsymbol{v}_{p_2}, \boldsymbol{v}_{p_3}, \boldsymbol{v}_q\}$ and $\{\boldsymbol{v}_{p_1}, \boldsymbol{v}_{p_2}, \boldsymbol{v}_{p_3}, \boldsymbol{v}_s\}$, respectively. These two tetrahedra (3-simplex) share a common base that is the triangle $\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}\boldsymbol{v}_{p_3}$ (2-simplex). We now observe that we can rotate one of the two tetrahedra, around the common base, until it is coincident with the 3D subspace in which the other tetrahedron lies. So we obtain a projection of the five points in a 3D Euclidean space, where all the inter-object distances are preserved except for the one between q and s for which we obtain either an upper and a lower bound.

The same intuition generalises into many dimensions. Assume a space with the $(n+2)$-point property, or better an Hilbert-embeddable space that has the n-point property for any n. We consider a set of n reference objects $\{p_1, \dots, p_n\}$, and two object q and s. We isometrically embed all the $(n+2)$ points into ℓ_2^{n+1}. Then we consider the $(n-1)$-simplex σ_0 formed by the vertex $\{\boldsymbol{v}_{p_1}, \dots, \boldsymbol{v}_{p_n}\}$. We call it *base* simplex (it correspond to the line $\overline{\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}}$ in the left-hand figure, and to the triangle $\boldsymbol{v}_{p_1}\boldsymbol{v}_{p_2}\boldsymbol{v}_{p_3}$ in the right-hand figure). As done in the previous cases we rotate the n-simplex σ_q of vertices $\{\boldsymbol{v}_{p_1}, \dots, \boldsymbol{v}_{p_n}, \boldsymbol{v}_q\}$ around the base simplex until it is coincident with the n-dimensional subspace in which the n-simplex σ_s of vertices $\{\boldsymbol{v}_{p_1}, \dots, \boldsymbol{v}_{p_n}, \boldsymbol{v}_s\}$ lies. Equivalently, we can rotate both σ_q and σ_s around σ_o until they are coincident with an arbitrary fixed n-dimensional subspace that contains σ_o. This subspace can be regarded as a n-dimensional Euclidean space in its on right. Note that there are two possible positions in ℓ_2^n for $\boldsymbol{v}_q$, one on either side of the hyperplane containing σ_0; we denote these as $\boldsymbol{v}_q^+$, and $\boldsymbol{v}_q^-$ respectively. Similarly, there are two possible positions in ℓ_2^n for $\boldsymbol{v}_s$, namely $\boldsymbol{v}_s^+$, and $\boldsymbol{v}_s^-$. As described in the next section, for a generic point a the vertices $\boldsymbol{v}_a^+$ and $\boldsymbol{v}_a^-$ can be computed just using the the distances between object a and all p_i. What we are going to prove is that

$$\ell_2^n(\boldsymbol{v}_s^+, \boldsymbol{v}_q^+) \quad \leq \quad d(s,q) \quad \leq \quad \ell_2^n(\boldsymbol{v}_s^+, \boldsymbol{v}_q^-). \tag{5.23}$$

Moreover, in Section 5.4.1 we present an algorithm to compute those vertices for which once $\boldsymbol{v}_a^+ = [x_1, \dots, x_n]$ is calculated, the vertex $\boldsymbol{v}_q^-$ is obtained at no extra cost as $[x_1, \dots, x_{n-1}, -x_n]$.

In conclusion, we present a new embedding technique that, using properties of finite isometric embedding, allows spaces with the n-point property to be translated in a smaller *Euclidean* space. In fact, we propose a family of function ϕ_n which can be created by measuring the distances among n objects sampled from the original space, and which can then be used to create a *surrogate* space:

$$\phi_n : (\mathcal{D}, d) \to (\mathbb{R}^n, \ell_2)$$

with the property

$$\ell_2\left(\phi_n(o_1), \phi_n(o_2)\right) \leq d(o_1, o_2) \leq g\left(\phi_n(u_1), \phi_n(u_2)\right), \tag{5.24}$$

where

$$g([x_1, \dots, x_n], [y_1, \dots, y_n]) = \sqrt{\sum_{i=1}^{n-1}(x_i - y_i)^2 + (x_n + y_n)^2}.$$

Note that the cost of evaluating the upper and the lower bound together is almost exactly the same as the cost of a ℓ_2 distance in $\mathbb{R}^n$.

The advantages of the proposed technique are that (a) the ℓ_2 metric is very much cheaper than some Hilbert-embeddable metrics; (b) the size of elements of $\mathbb{R}^n$ may be much smaller than elements of $\mathcal{D}$ (dimensionality reduction), and (c) in many cases we can achieve both of these along with an *increase* in the scalability of the resulting search space.

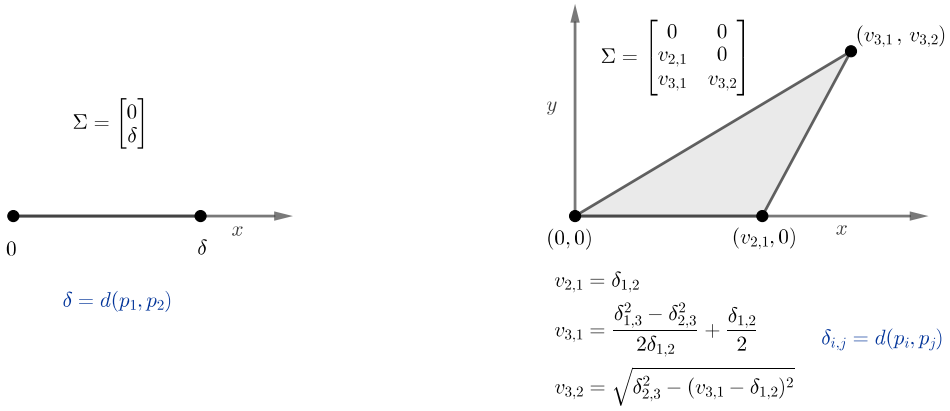


Figure 5.26: *Examples of n-simplexes built starting from $n = 2$ objects p_1, p_2 (left-hand figure), and $n = 3$ objects p_1, p_2, p_3 (right-hand figure).*

5.4.1 Constructing Simplexes from Edge Length

We recall that a *n-simplex* σ spanned by the points $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n+1} \in \ell_2^n$ is the set

$$\{\mathbf{x} = \sum_{i=1}^{n+1} t_i \boldsymbol{v}_i \mid \sum_{i=1}^{n+1} t_i = 1, t_i \geq 0\}.$$

The points $\boldsymbol{v}_i$ are called *vertices* of σ; n is called the *dimension* of σ. Any simplex spanned by a subset of $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n+1}\}$ is called a *face* of σ. The structure of a simplex in n-dimensional space is given as an $n + 1$ by n matrix Σ representing the Cartesian coordinates of each vertex. For example, the following matrix represents four coordinates which are the vertices of a tetrahedron in 3D space:

$$\begin{bmatrix} 0 & 0 & 0 \\ v_{2,1} & 0 & 0 \\ v_{3,1} & v_{3,2} & 0 \\ v_{4,1} & v_{4,2} & v_{4,3} \end{bmatrix}$$

For all such matrices Σ, the invariant that $v_{i,j} = 0$ whenever $j \geq i$ can be maintained without loss of generality; for any simplex, this can be achieved by rotation and translation within the Euclidean space while maintaining the distances among all the vertices. Furthermore, if we restrict $v_{i,j} \geq 0$ whenever $j = i - 1$ then in each row this component represents the *altitude* of the i^{th} point with respect to a base face represented by the matrix cut down from Σ by selecting elements above and to the left of that entry.

In our context, we are interested in construct a $(n - 1)$-simplex in ℓ_2^n based only on the distances measured among n reference points of $\mathcal{D}$. We then use it as a base simplex to construct new apexes above it, which are the objects of our "surrogate space". Figure 5.26 shows base cases of simplexes constructed from two and three reference points, respectively.

5.4.1.1 Simplex and Apex Construction

In [78], we proposed an algorithm for determining Cartesian coordinates for the vertices of a simplex, given *only* the distances between points. Our algorithm is inductive, at each stage allowing the apex of an n-dimensional simplex to be determined given the coordinates of an $(n - 1)$-dimensional simplex, and the distances from the new apex to each vertex in the existing simplex (Algorithm 4). This is important because, given a fixed base simplex (with n vertices) over which many new apexes are to be constructed, the time required to compute each one is $\mathcal{O}(n)$ distance computations. The total cost in term of distance calculations to construct a simplex with $(n + 1)$ vertices in n dimensions is $\mathcal{O}(n^2)$.

Algorithm 4: nSimplexBuild

Input: n points $p_1, \ldots, p_n \in (\mathcal{D}, d)$
Output: $(n-1)$-dimensional simplex in ℓ_2^{n-1} represented by the matrix $\Sigma \in \mathbb{R}^{n \times (n-1)}$

$\Sigma \leftarrow 0 \in \mathbb{R}^{n \times (n-1)}$;
if $n = 2$ **then**
  $\delta \leftarrow d(p_1, p_2)$;
  $\Sigma \leftarrow \begin{bmatrix} 0 \\ \delta \end{bmatrix}$;
  return Σ;
end
$\Sigma_{Base} \leftarrow$ nSimplexBuild($p_1, \ldots, p_{n-1}$); `// (n-2)-dimensional simplex`
$Distances \leftarrow 0 \in \mathbb{R}^{n-1}$;
for $i = 1$ **to** $n-1$ **do**
  $Distances[i] \leftarrow d(p_i, p_n)$;
end
$newApex \leftarrow$ ApexAddition($\Sigma_{Base}, Distances$);
for $1 \leq i \leq n$ and $1 \leq j \leq i-1$ set $\Sigma[i][j]$ to $\Sigma_{Base}[i][j]$;
for $1 \leq j \leq n$ set $\Sigma[n+1][j]$ to $newApex[j]$;
return Σ;

Algorithm 5: ApexAddition

Input: A $(n-1)$-dimensional base simplex (Σ_{Base}), and the distances of a new (unknown) apex point ($Distances$) from the vertices of the base simplex:

$$\Sigma_{\text{Base}} = \begin{bmatrix} 0 & & & \\ v_{2,1} & 0 & & \mathbf{0} \\ v_{3,1} & v_{3,2} & \ddots & \\ \vdots & & \ddots & 0 \\ v_{n,1} & & \cdots & v_{n,n-1} \end{bmatrix} \in \mathbb{R}^{n \times (n-1)}$$

$$Distances = \begin{bmatrix} \delta_1 & \cdots & \delta_n \end{bmatrix} \in \mathbb{R}^n$$

Output: The cartesian coordinates of the new apex point

$Output \leftarrow \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^n$;
for $i = 2$ **to** n **do**
  $l \leftarrow \ell_2(\Sigma_{Base}[i], Output)$;
  $\delta \leftarrow Distances[i]$;
  $x \leftarrow \Sigma_{Base}[i][i-1]$;
  $y \leftarrow Output[i-1]$;
  $Output[i-1] \leftarrow y - (\delta^2 - l^2)/2x$;
  $Output[i] \leftarrow \sqrt{y^2 - (Output[i-1])^2}$;
end
return $Output$

For $n+1$ data points, where $n > 1$, an $(n-1)$-dimensional simplex is first constructed using the distances among the first n points. This simplex is used as a simplex base to which a new apex, the $(n+1)^{th}$ point, is added by the *ApexAddition* algorithm (Algorithm 5). In essence, the *ApexAddition* algorithm is derived from exactly the same intuition as the lower-bound property explained earlier, at each stage lifting the final dimension out of the same hyperplane into a new dimension to capture the measured distances.

Lemma 5.4.1 (Correctness of the ApexAddition algorithm)**.** *Let $\Sigma_{Base} \in \mathbb{R}^{n \times n-1}$ representing a $(n-1)$-dimensional simplex of vertices $\Sigma_{Base}[i] \in \ell_2^{n-1}$, with $\Sigma_{Base}[i][j] = 0$ for all $j \geq i$ and $\Sigma_{Base}[n][n-1] \geq 0$. Let $\boldsymbol{v}_i$ the corresponding vertices in ℓ_2^n (obtained from $\Sigma_{Base}[i]$ by adding a zero to the end of the vector) and let δ_i the distance between an unknown apex point and the vertex $\boldsymbol{v}_i$. Let $\boldsymbol{o} = \begin{bmatrix} o_1 & \dots & o_n \end{bmatrix}$ the output of the* ApexAddition *Algorithm. Then $\boldsymbol{o}$ is a feasible apex, i.e. it is a point in $\mathbb{R}^n$ satisfying $\ell_2(\boldsymbol{o}, \boldsymbol{v}_i) = \delta_i$ for all $1 \leq i \leq n$. The last component o_n is non-negative and represents the* altitude *of $\boldsymbol{o}$ with respect to the base face Σ_{Base}.*

Proof. It is sufficient to prove that the output $\boldsymbol{o} = \begin{bmatrix} o_1 & \dots & o_n \end{bmatrix}$ of the Algorithm 5 has distance δ_i from the vertex $\boldsymbol{v}_i$,*i.e.* satisfies the following equations

$$\begin{cases} o_1^2 + \dots + o_n^2 = \delta_1^2 & (5.25.1) \\ \quad \vdots & \\ \sum_{j=1}^{i-1}(v_{i,j} - o_j)^2 + \sum_{j=i}^{n} o_j^2 = \delta_i^2 & (5.25.i) \\ \quad \vdots & \\ \sum_{j=1}^{n-1}(v_{n,j} - o_j)^2 + o_n^2 = \delta_n^2 & (5.25.n) \end{cases} \tag{5.25}$$

Note that the i-th component of the output $\boldsymbol{o}$ is updated only at the iteration i and $i+1$ of the *ApexAddition* Algorithm (lines 7-8). So, if we denote with $\boldsymbol{o}^{(i)}$ the output at the end of iteration i we have:

$$\boldsymbol{o}^{(1)} = \begin{bmatrix} \delta_1 & 0 & \dots & 0 \end{bmatrix}, \quad \boldsymbol{o} = \boldsymbol{o}^{(n)} \tag{5.26}$$

$$o_i = o_i^{(h)}, \quad o_h^{(i)} = 0 \qquad 1 \leq i < h \leq n \tag{5.27}$$

$$o_{i-1}^{(i)} = o_{i-1}^{(i-1)} - \frac{\delta_i^2 - \ell_2(\boldsymbol{v}_i, o^{(i-1)})}{2v_{i,i-1}} \qquad 2 \leq i \leq n \tag{5.28}$$

$$(o_i^{(i)})^2 = (o_{i-1}^{(i-1)})^2 - (o_{i-1}^{(i)})^2 \qquad 1 \leq i \leq n-1 \tag{5.29}$$

and thus

$$o_{i-1} = o_{i-1}^{(i-1)} - \frac{\delta_i^2 - \sum_{j=1}^{i-2}(v_{i,j} - o_j)^2 - (v_{i,i-1} - o_{i-1}^{(i-1)})^2}{2v_{i,i-1}} \qquad 2 \leq i \leq n \tag{5.30}$$

$$(o_{i-1})^2 = (o_{i-1}^{(i-1)})^2 - (o_i^{(i)})^2 \qquad 1 \leq i \leq n-1 \tag{5.31}$$

By combining Eq. (5.27) and (5.31) we obtain

$$\sum_{j=i}^{n} o_j^2 = (o_i^{(i)})^2 \qquad 1 \leq i \leq n-2, \tag{5.32}$$

and so Eq. (5.25.1) clearly holds. Moreover, from Eq. (5.32) and Eq. (5.30), it follows that $\boldsymbol{o}$ satisfies Eq. (5.25.i) for all $i = 2, \dots, n$:

$$\sum_{j=1}^{i-1}(v_{i,j} - o_j)^2 + \sum_{j=i}^{n} o_j^2 \overset{(5.32)}{=} v_{i,i-1}^2 - 2v_{i,i-1}\, o_{i-1} + \sum_{j=1}^{i-2}(v_{i,i-1} - o_j)^2 + (o_{i-1}^{(i-1)})^2 \overset{(5.30)}{=} \delta_i^2$$

□

5.4.2 n-Simplex projection and Distances Bounds

Consider a metric space $(\mathcal{D}, d)$ which is isometrically n-embeddable in ℓ_2^{n-1}, for any $n > 1$. It is possible to pick any n reference points $p_1, \ldots, p_n$ and form a simplex Σ_n in ℓ_2^{n-1}. For any two further points q and s, two further simplexes based on Σ_n can be independently calculated in ℓ_2^n space, using finite mappings $q \rightarrow \boldsymbol{q}^{(n)}$ and $s \rightarrow \boldsymbol{s}^{(n)}$, where $\boldsymbol{q}^{(n)}$ and $\boldsymbol{s}^{(n)}$ are computed using the *ApexAddition* Algorithm. Let $\phi_n : (\mathcal{D}, d) \rightarrow \ell_2^n$ the projection that using the base simplex Σ_n associates the apex $\boldsymbol{s}^{(n)}$ to the object s. For an arbitrary set of objects $s_i \in \mathcal{S}$, the apex $\phi_n(s_i)$ can be pre-calculated. Because of the method we use to build simplexes, the final coordinate always represents the altitude of the new apex above the hyperplane containing the base simplex. Given this, two apexes exist, according to whether a positive or negative real number is inserted at the final step of the algorithm. We are going to prove that given the apexes

$$\begin{aligned} \phi_n(s) &= [x_1, x_2, \ldots, x_{n-1}, x_n] \\ \phi_n(q) &= [y_1, y_2, \ldots, y_{n-1}, y_n] \end{aligned}$$

then

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad \leq \quad d(s,q) \quad \leq \quad \sqrt{\sum_{i=1}^{n-1}(x_i - y_i)^2 + (x_n + y_n)^2}$$

From the structure of these calculations, it is apparent that they are likely to converge rapidly around the true distance as the number of dimensions used becomes higher, as we show in following Lemma 5.4.2. It can also be seen that the cost of calculating both of these values together, especially in higher dimensions, is essentially the same as a simple ℓ_2 calculation. The main outcome is that the Euclidean distance ℓ_2 can be used as a filter function for d. When a query is performed, only n distances in the metric space require to be calculated to discover the new apex $\phi_n(q)$ in ℓ_2^n. Whenever it may be deduced, via an indexing structure or otherwise, that $\ell_2^n(\phi_n(s), \phi_n(q)) > t$, then s cannot be a solution to the query. If ℓ_2^n is substantially cheaper to calculate than d, there are various ways of using this property for efficient query evaluation. Note that in the case where $n = 1$, the above gives the general property of pivoting as used in metric spaces. When $n = 2$, then it corresponds to the tetrahedral/planar projection. In this sense, our observations amount to a generalisation of these indexing properties into arbitrary dimensional Euclidean space.

Finally, we note that the lower-bound function is a proper metric, but the upper-bound function is not even a semi-metric: even although it is a Euclidean distance in the apex space, one of the domain points is constructed by reflection across a hyperplane and thus the distance between a pair of identical points is in general non-zero.

Lemma 5.4.2 (n-Simplex Distance Constraint)**.** *Let $(\mathcal{D}, d)$ a space isometrically $(n+2)$-embeddable in ℓ_2^{n+1}. Let $p_1, \ldots, p_n \in \mathcal{D}$ and, for any $k \leq n$, let σ_k the $(k-1)$-dimensional simplex generated from $p_1, \ldots, p_k$ by using the* nSimplexBuild *Algorithm. For any $o \in \mathcal{D}$, let $\boldsymbol{o}^{(k)} \in \ell_2^k$ the apex point with distance $d(o, p_1), \ldots, d(o, p_k)$ from the vertices of σ_k, computed using the* ApexAddition *Algorithm. Then for all $q, s \in U$,*

1. $\ell_2^{k-1}(\boldsymbol{s}^{(k-1)}, q^{(k-1)}) \leq \ell_2^k(\boldsymbol{s}^{(k)}, \boldsymbol{q}^{(k)})$ *for* $2 \leq k \leq n$
2. $g(\boldsymbol{s}^{(k-1)}, \boldsymbol{q}^{(k-1)}) \geq g(\boldsymbol{s}^{(k)}, \boldsymbol{q}^{(k)})$ *for* $2 \leq k \leq n$
3. $\ell_2^n(\boldsymbol{s}^{(n)}, \boldsymbol{q}^{(n)}) \leq d(s,q) \leq g(\boldsymbol{s}^{(n)}, \boldsymbol{q}^{(n)})$

where, for any $k \in \mathbb{N}$, $g : \ell_2^k \rightarrow \ell_2^k$ is defined as $g(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{k-1}(x_i - y_i)^2 + (x_k + y_k)^2}$.

Proof. By construction, for any $k \leq n$ and any object o we have

$$o_i^{(k)} = o_i^{(k-1)} \qquad i = 1, \ldots, k-2 \tag{5.33}$$

$$o_i^{(i)} \geq 0 \qquad i = 1, \ldots, k \tag{5.34}$$

$$(o_{k-1}^{(k)})^2 + (o_k^{(k)})^2 = (o_{k-1}^{(k-1)})^2 \tag{5.35}$$

Condition 1 directly follows from Eq. (5.33)-(5.35):

$$\begin{aligned}\ell_2^k(\boldsymbol{s}^{(k)},\boldsymbol{q}^{(k)})^2 &= \ell_2^{k-1}(\boldsymbol{s}^{(k-1)},\boldsymbol{q}^{(k-1)})^2 - (s_{k-1}^{(k-1)} - q_{k-1}^{(k-1)})^2 + \sum_{i=k-1}^{k}(s_i^{(k)} - q_i^{(k)})^2 \\ &= \ell_2^{k-1}(\boldsymbol{s}^{(k-1)},\boldsymbol{q}^{(k-1)})^2 + 2\Big[- s_{k-1}^{(k)}q_{k-1}^{(k)} - s_k^{(k)}q_k^{(k)} \\ &\qquad + \sqrt{(s_{k-1}^{(k)})^2 + (s_k^{(k)})^2}\sqrt{(q_{k-1}^{(k)})^2 + (q_k^{(k)})^2}\Big] \\ &\geq \ell_2^{k-1}(\boldsymbol{s}^{(k-1)},\boldsymbol{q}^{(k-1)})^2\end{aligned}$$

where the last passage follows from the Cauchy–Schwarz inequality [7].

Similarly, Condition 2 also holds:

$$\begin{aligned}g(\boldsymbol{s}^{(k)},\boldsymbol{q}^{(k)})^2 &= g(\boldsymbol{s}^{(k-1)},\boldsymbol{q}^{(k-1)})^2 + 2\Big[- s_{k-1}^{(k)}q_{k-1}^{(k)} + s_k^{(k)}q_k^{(k)} \\ &\qquad - \sqrt{(s_{k-1}^{(k)})^2 + (s_k^{(k)})^2}\sqrt{(q_{k-1}^{(k)})^2 + (q_k^{(k)})^2}\Big] \\ &\leq g(\boldsymbol{s}^{(k-1)},\boldsymbol{q}^{(k-1)})^2.\end{aligned}$$

Now we prove that $\ell_2^n(\boldsymbol{s}^{(n)},\boldsymbol{q}^{(n)})$ and $g(\boldsymbol{s}^{(n)},\boldsymbol{q}^{(n)})$ are, respectively, a lower bound and an upper bound for the actual distance $d(s,q)$. The main idea is using the simplex σ_n spanned by $p_1,\dots,p_n$ as a base face to build the simplex σ_{n+1} spanned by $p_1,\dots,p_n,s$ and then use the latter as base face to build the simplex σ_{n+2} spanned by $p_1,\dots,p_n,s,q$. In this way, we have an isometric embedding of $p_1,\dots,p_n,s,q$ into ℓ_2^{n+1} that is the function that maps $p_1,\dots,p_n,s,q$ into the vertices of σ_{n+2}. So, given the base simplex σ_n (represented by the matrix Σ_n), and the apex $\boldsymbol{s}^{(n)},\boldsymbol{q}^{(n)} \in \ell_2^n$ we have that the simplex σ_{n+2} is represented by

$$\Sigma_{n+2} = \left[\begin{array}{ccc|cc} & \Sigma_n & & & 0 \\ \hline s_1^{(n)} & \cdots & s_{n-1}^{(n)} & s_n^{(n)} & 0 \\ q_1^{(n)} & \cdots & q_{n-1}^{(n)} & q_n^{(n+1)} & q_{n+1}^{(n+1)} \end{array}\right] \in \mathbb{R}^{(n+2)\times(n+1)} \tag{5.36}$$

where, by construction, $(q_{n+1}^{(n+1)})^2 = (q_n^{(n)})^2 - (q_n^{(n+1)})^2$, $s_n^{(n)}, q_{n+1}^{(n+1)} \geq 0$, and $d(q,s)$ equals the Euclidean distance between the two last rows of Σ_{n+2}.

It follows that

$$d(q,s)^2 = \sum_{i=1}^{n-1}(s_i^{(n)} - q_i^{(n)})^2 + (s_n^{(n)})^2 + (q_n^{(n)})^2 - 2s_n^{(n)}q_n^{(n+1)}; \tag{5.37}$$

and, since $q_n^{(n)} \geq |q_n^{(n+1)}|$, we have

$$d(q,s)^2 = \ell_2^n(\boldsymbol{s}^{(n)},\boldsymbol{q}^{(n)})^2 + 2s_n^{(n)}(q_n^{(n)} - q_n^{(n+1)}) \geq \ell_2^n(\boldsymbol{s}^{(n)},\boldsymbol{q}^{(n)})^2,$$

and

$$d(q,s)^2 = g(\boldsymbol{s}^{(n)},\boldsymbol{q}^{(n)})^2 - 2s_n^{(n)}(q_n^{(n)} + q_n^{(n+1)}) \leq g(\boldsymbol{s}^{(n)},\boldsymbol{q}^{(n)})^2.$$

□

[7] Cauchy–Schwarz inequality in two dimension is: $(a_1b_1 + a_2b_2)^2 \leq (a_1^2 + a_2^2)(b_1^2 + b_2^2)\ \forall a_1, b_1, a_2, b_2 \in \mathbb{R}$, which implies

$$(a_1b_1 + a_2b_2) \leq \sqrt{(a_1^2 + a_2^2)}\sqrt{(b_1^2 + b_2^2)} \quad \forall a_1, b_1, a_2, b_2 \in \mathbb{R}$$

5.4.3 Experiments

We now investigate the use of the n-Simplex projection. Since the lower-bound is a metric, in Section 5.4.3.1 we perform some tests to measure its distortion with respect to the original distance. We then show results obtained using the n-simplex projection for the indexing and searching task (Section 5.4.3.2). Finally, in Section 5.4.3.3, we "visualize" the convergence of n-simplex upper and lower bounds using real-word high dimensional vectors.

5.4.3.1 Measuring Distortion

In Section 2.4.8, we introduced the concept of *distortion* of a metric space transformation $f : (\mathcal{D}, d) \rightarrow (\mathcal{D}', d')$ as the smallest value D such that, for some scaling factor r

$$\forall o_1, o_2 \in \mathcal{D}, \qquad d'(f(o_1), f(o_2)) \quad \leq \quad d(o_1, o_2) \quad \leq \quad D \cdot r \cdot d'(f(o_1), f(o_2)).$$

An approximation $(\mathcal{D}', d')$ of the space $(\mathcal{D}, d)$ with distortion equals to one is practically equivalent to the space $(\mathcal{D}, d)$ for metric search tasks.

We have measured the distortion for a number of different transformation functions and metric measures. In particular, we compared the n-Simplex projection with the following "dimensionality reduction" transformations:

PCA - The *Principal Component Analysis* [148] is a popular technique for unsupervised dimensionality reduction in vector spaces. It provides a linear transformation of k-dimensional to n-dimensional *euclidean vectors* ($n \leq k$) that best preserves the *variance* of the input data. Specifically, PCA projects the data along the direction of its first n principal components, which are the eigenvectors of the covariance matrix of the (centered) input data.

JL - According to the *Johnson-Lindenstrauss Flattening Lemma* (see e.g. [178, pag. 358]) it is possible to linearly embed a finite set of m *euclidean vectors* of ℓ_2^k into a n-dimensional Euclidean space ($n < m$) with a "small" distortion. Specifically the Lemma asserts that for any m-points of ℓ_2^k and every $0 < \epsilon < 1$ there is a mapping into ℓ_2^n that preserves all the interpoint distances within factor $1 + \epsilon$, where $k = O(\epsilon^{-2} \log m)$. The embedding given in the original proof of Johnson Lindenstrauss lemma [147] is particularly simple to implement by using a random matrix of size $n \times k$ where rows are chosen as a random n-tuple of orthonormal vector in $\mathbb{R}^k$.

LMDS - For general *metric space*, perhaps the best known projection technique is the *metric Multidimensional Scaling (MDS)*, which aims to preserve *inter-point distances*. Given m objects and the distances $\delta_{i,j}$ between those objects, MDS finds a set on m points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ in a Euclidean space $\mathbb{R}^n$ such that $\ell_2^n(\mathbf{x}_i, \mathbf{x}_j)$ is close as possible to $\delta_{i,j}$. The coordinates of the point $\mathbf{x}_i$ are computed using a spectral analysis of the matrix of the squared interpoint distances. When the number m of data points is large the classical MDS is too expensive in practice due to a requirement for $O(m^2)$ distance computations and spectral decomposition of a $m \times m$ matrix. The *Landmark MDS (LMDS)* [85] is a fast approximation of MDS. LMDS uses a set of n *landmark* points (pivots) to compute $n \times m$ distances of the data points from the pivots. It applies classical MDS to these points and uses a distance-based triangulation procedure to project the remaining data points [8].

The PCA and JL can be used only in Euclidean space, the LMDS can be used in any metric space, while our n-Simplex projection can be used in metrics with the n-point property. Moreover, we observe that in LMDS the use of a projection learned on some landmark points to project the rest of the input data has many analogies with our n-Simplex projection.

Summary results on SISAP *Colors* benchmark are shown in Figure 5.27 for both Euclidean (left-hand figure) and Jenson-Shannon (right-hand figure) distance. In each case, the X-axis represents the number of dimensions used for the representation, with the distortion plotted against this. For Euclidean distance, there are two entries for n-Simplex: one for randomly-selected reference points, and the other

[8] If $X_n \in \mathbb{R}^{n \times n}$ is the output of MDS on the pivot set, the embedding of a new point s into $\mathbb{R}^n$ is computed as $\mathbf{x}_s = -\frac{1}{2} X_n^+ (\boldsymbol{\delta}_s^2 - \boldsymbol{\delta}_\mu^2)$, where $(\delta_s^2)_i = d(s, p_i)$, $(\delta_\mu^2)_j = \frac{1}{n} \sum_{i=1}^n d(p_i, p_j)$, and X_k^+ is the pseudo inverse of X_n.

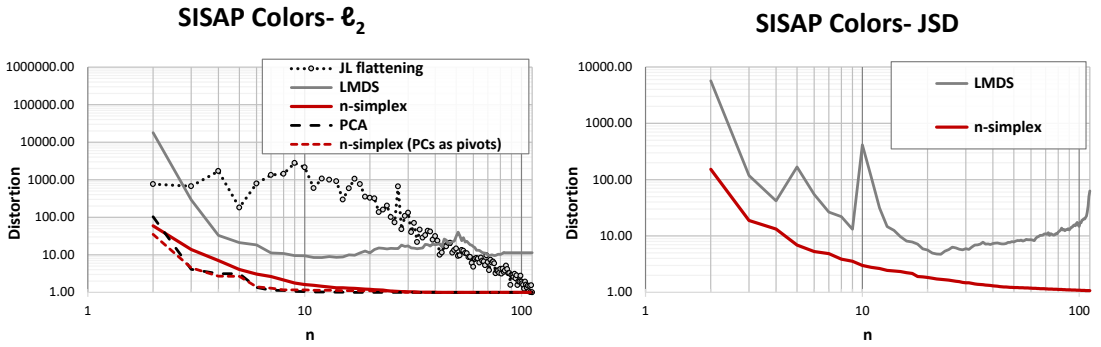


Figure 5.27: *Distortion measurements for various dimensionality reduction strategies for the SISAP* Colors *dataset. The left figure gives measurements for Euclidean distance, the right for Jensen-Shannon distance where only LMDS and n-Simplex are applicable. The* Colors *dataset has 112 physical dimensions.*

where the choice of reference points is guided by the use of PCA. In the latter case we select the first n principal components (eigenvectors of the covariance matrix) as pivots.

It can be seen that n-Simplex outperforms all other strategies except for PCA, which is not applicable to non-Euclidean spaces. LMDS is the only other mechanism applicable to general metric spaces[9]; this is a little more expensive than n-Simplex to evaluate, and performs relatively badly. The comparison with JL is a slightly unfair, as the JL lemma applies only for very high dimensions in an evenly distributed space; we have also tested such spaces, and JL is still outperformed by n-Simplex, especially at lower dimensions.

The distortion we show here is only for the lower-bound function of n-Simplex. We have measured the upper-bound function also, which gives similar results. Unlike the lower-bound, the upper-bound is not a proper metric; however, for non-metric approximate search it should be noted that the mean of the lower- and upper-bound functions give around half the distortion plotted here.

The implications of these results for exact search should be noted. For Euclidean search, the distortion has dropped to almost one at between 20 and 30 dimensions, implying the possibility of accurate search using data which is less than one-quarter of the original size. For Jensen-Shannon, more dimensions will be required, but the cost of the ℓ_2 metric required to search the compressed space is around one-hundredth the cost of the original metric. In the next section, we present experimental results consistent with these observations.

5.4.3.2 Exact search: Indexing with n-Simplex

The Euclidean metric on the surrogate space obtained with the n-simplex projection is a lower bounding of the original distance, so exact search for any range query can be performed in the surrogate space in order to select a set of candidate results to be subsequently refined using the original distance. Moreover, the upper-bound can also be exploited during the search to avoid unnecessary (original) distance evaluations.

An exact search mechanism relied on the n-Simplex projection can be viewed as similar to LAESA (see Section 2.4.7.5) in that there exists an underlying data structure which is a table of numbers, n per original object, with the intention of using this table to exclude candidates which cannot be within a given search threshold.

Figure 5.28 shows an illustration of the tables used in LAESA and in n-Simplex search. In both cases, n reference objects are chosen from the space and used to represent the data objects. For LAESA, each

[9] In [85] the authors note it works better for some metrics than for others; in our understanding, it will work well only for spaces with the n-point property.

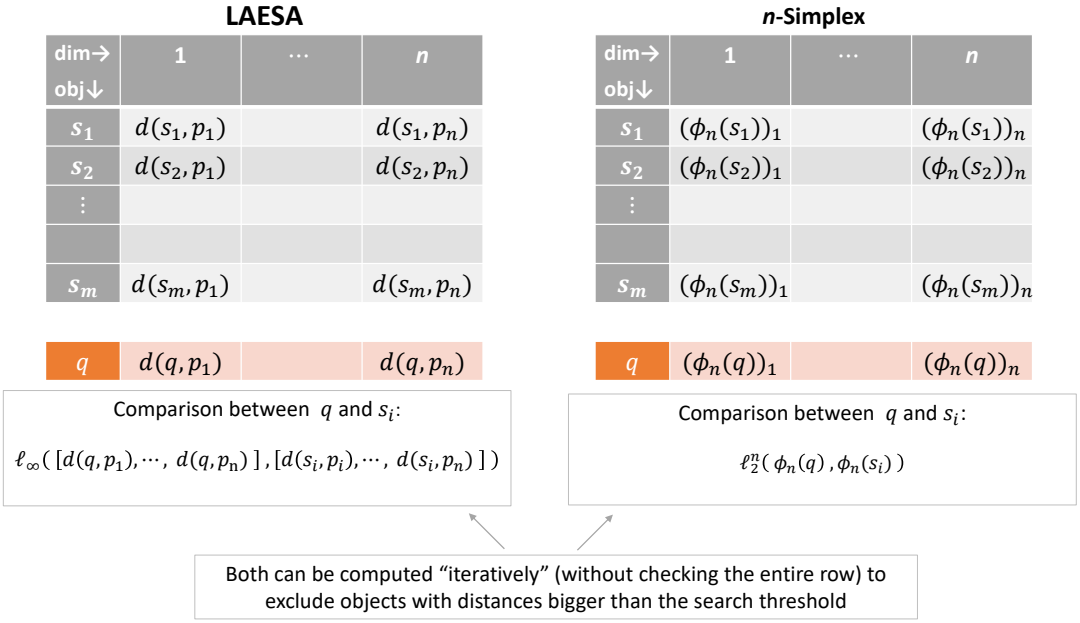


Figure 5.28: *Example of tables built for LAESA and n-Simplex to search a set of m objects. In the depicted example, n reference objects are used to built both the representations.*

row of the table is filled with the distances from a data object to all the reference points. For n-Simplex, each row is filled with the n Cartesian coordinates of the apex $\phi_n(s)$ built over the $(n-1)$-dimensional simplex formed from the reference points. We remind that the apex is calculated only using the distances of the data object from the reference points. At query time, the distances from the query to each reference object are calculated. In the case of LAESA, the metric for comparison is Chebyshev (ℓ_∞): that is, if any pairwise difference $|d(s_i,p_j) - d(q,p_j)|$ is greater than the query threshold, the object s_i from which that row was derived cannot be a solution to the query. For n-Simplex, the metric used is ℓ_2: that is if the apex represented in a row is further than the query threshold from the apex generated from the query, again the object from which that apex was derived cannot be a solution to the query.

In both cases, there are two ways of approaching the table search. It can be performed sequentially over the whole table, in which case either metric can be terminated within a row if the threshold is exceeded, without continuing to the end of the row. Alternatively, the table can itself be reindexed using a metric index. Although this compromises the amount of space available for the table itself, it may avoid many of the individual row comparisons. We also note that our surrogate space has the four-point property since it is a Euclidean space, so in the context of re-indexing the Hilbert Exclusion property may be used.

In all cases, the result is a filtered set of candidate objects which is guaranteed to contain the correct solution set. In general, this set must be re-checked against the original metric, in the original space. For n-Simplex the upper-bound condition is checked first; if this is less than the query threshold, then the object is guaranteed to be an element of the result set with no further check required.

Any such mechanism will perform differently over datasets with different characteristics. To give useful comparisons with other studies in the literature, we apply the techniques to the SISAP *Colors* [101], using three different supermetrics: Euclidean, Cosine, and Jensen-Shannon. We chose this dataset because (a) it has only positive values and is therefore indexable by all of the considered metrics, and (b) it shows an interesting non-uniformity, in that its intrinsic dimensionality [66] for all metrics is much less than its physical dimensionality (112). It should thus give an interesting "real world" context to assess the relative value of the different mechanisms.

Results As done in the experiments of the previous sections, when using the Euclidean distance to search the SISAP *Colors*, we used three benchmark thresholds that return the 0.01%, 0.1%, and the 1% of the data, respectively. For the Jenson-Shannon and Cosine metrics, we chose thresholds that return around 0.01% of the data. In all cases, the first 10% of the objects are used to query the remaining 90%. Pivots are randomly-selected both for LAESA and n-Simplex approach.

For each metric, we tested different mechanisms with different allocations of space: 5 to 50 numbers per data element, thus the space used per object is between 4.5% and 45% of the original. All results reported are for exact search, that is the initial filtering is followed by re-testing within the original space where required. Five different mechanisms were tested, as follows:

sequential LAESA each row of the table is scanned sequentially, each element of each row is tested against the query and that row is abandoned if the absolute difference is greater than the threshold.

reindexed LAESA the data in the table is indexed using a Monotonous Bisector Tree (MBT), searched using the Chebyshev metric and hyperbolic exclusion [10].

sequential n-Simplex each row of the table is scanned sequentially, for each element of each row the square of the absolute difference is added to an accumulator, the row is abandoned if the accumulator exceeds the square of the threshold, and the upper-bound is applied if the end of the row is reached before re-checking in the original space.

reindexed n-Simplex the data in the table is indexed using a MBT using the Hilbert exclusion property, and searched using the Euclidean metric; the upper-bound is applied for all results, before re-checking in the original space.

MBT the original space is indexed using a MBT with the Hilbert exclusion property.

The MBT is used as, in the previous section, this has been found to be the best-performing *simple* indexing mechanism for use with Hilbert Exclusion. Each approach returns two lists of object identifier: 1) identifiers of objects that are known to be in the solution set, 2) identifiers of "candidate" objects that need to be checked against the original metric space. Therefore, in all cases the results set is exact.

For each mechanism we measure the elapsed time, the number of original-space distance calculations performed and, in the case of the re-indexing mechanisms, the number of reindexed space calculations. Please note that the reindexed space calculations are relative to ℓ_∞ for LAESA and ℓ_2 for n-Simplex. Also in this case the all the used code is available at the Metric Space Framework [6] for independent testing. The tests were run by Prof. Richard Connor on a 2.8 GHz Intel Core i7, running on an otherwise bare machine without network interference. The code is written in Java, and all datasets used fit easily into the Java heap without paging or garbage collection occurring.

Table 5.5 shows the elapsed time for Euclidean distance, also reported in Figure 5.29 in graphical form. Interestingly, reindexed n-Simplex consistently and significantly outperforms normal MBT at between 15 and 25 dimensions, depending on the query threshold. It is also interesting to see that, as the query threshold increases, and therefore scalability decreases, the sequential n-Simplex takes over as the most efficient mechanism, again with a "sweet spot" at 15 dimensions.

Table 5.6 and Figure 5.30 show the same experiment performed with Cosine and Jensen-Shannon distances. In the case of Jenson-Shannon, the extra relative cost saving from the more expensive metrics is very clear, with relative speedups of 4.5 and 8.5 times respectively. In the Jensen-Shannon tests, the relatively very high cost of the metric evaluation to some extent masks the difference between sequential and reindexed n-Simplex, but we note that the former maintains scalability while the latter does not.

Finally we report the actual number of distance measurement made for Euclidean (Table 5.7), Cosine and Jensen-Shannon (Table 5.8) searches. We report the number of distance calls required in both the original and reindexed spaces. Note that original-space calls are the same for both sequential and reindexed mechanisms and include the distance computations needed to refine the candidate result set and the distance calculations between objects and reference points. The index calls are the number of metric calls made during the index search, so ℓ_2 distances for the n-Simplex and ℓ_∞ for LAESA. Interestingly

[10] The Hilbert exclusion cannot be used with Chebyshev metric.

Table 5.5: *Elapsed Times - SISAP* Colors*, Euclidean distance.*
All times are in seconds, for executing 11268 queries over 101414 data. The MBT performance are independent from the number n and the value reported is the mean obtained over several running. L_{seq} is the sequential LAESA. L_{rei} is the reindexed LAESA. N_{seq} is the sequential n-Simplex. N_{rei} is the reindexed n-Simplex.

	$t_{0.01\%} = 0.051768$					$t_{0.1\%} = 0.082514$					$t_{1\%} = 0.131163$				
Dims	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*MBT*	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*MBT*	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*MBT*
5	18.6	28.0	13.8	5.8	5.5	33.4	80.9	22.4	29.0	18.1	56.2	201.6	34.9	70.4	54.4
10	17.7	22.1	15.0	3.3		30.3	67.9	20.3	14.7		58.1	220.3	25.5	50.6	
15	16.3	15.2	14.6	**3.0**		26.7	59.7	20.2	12.1		45.8	159.5	**24.4**	44.7	
20	19.0	16.3	18.9	3.3		28.2	56.6	19.4	**11.5**		46.8	189.3	27.8	48.3	
25	22.5	16.9	20.4	3.4		27.4	56.8	22.3	13.4		45.5	167.5	26.2	40.1	
30	20.9	16.8	20.4	3.5		28.6	57.3	24.5	13.6		45.9	181.2	28.5	45.1	
35	22.0	16.4	21.3	3.9		28.7	65.0	22.5	13.9		43.9	163.0	31.2	44.9	
40	23.1	17.3	22.1	4.0		28.8	55.9	22.8	14.3		49.4	180.5	34.2	46.1	
45	22.5	18.7	22.2	4.4		32.0	61.5	27.7	15.0		48.5	169.8	37.1	44.9	
50	21.3	17.1	18.9	4.5		32.0	59.0	24.0	15.5		55.2	207.6	34.5	45.3	

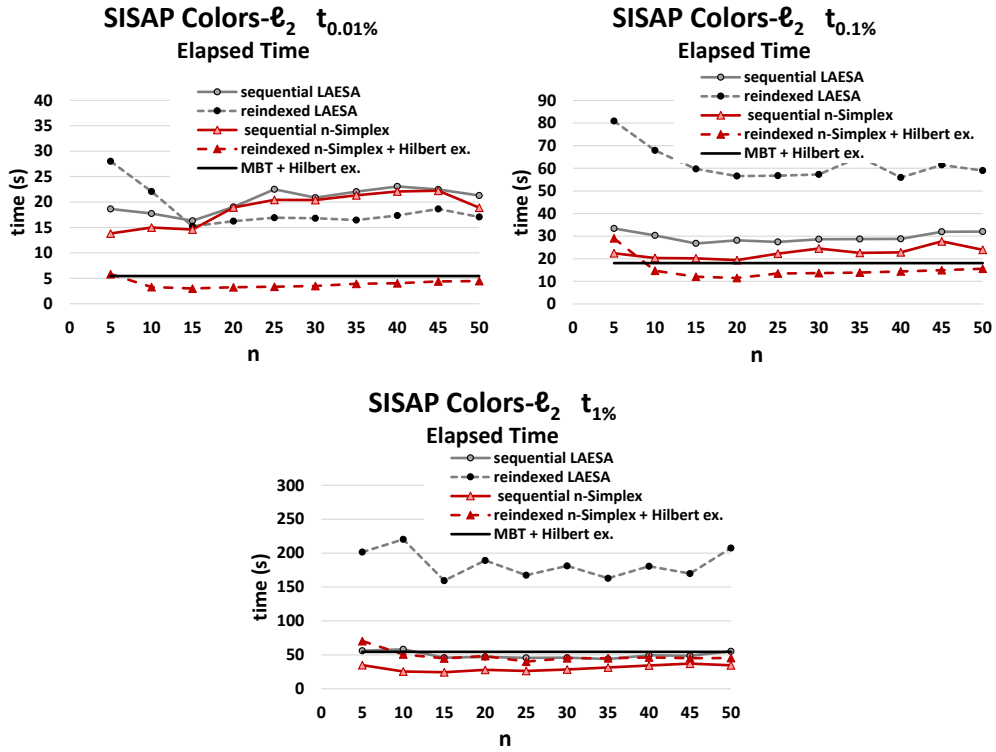


Figure 5.29: *Elapsed Times - SISAP Colors, Euclidean metric. (Results from Table 5.5).*

Table 5.6: *Elapsed Times - SISAP* Colors*, with Cosine and Jensen-Shannon distances. All times are in seconds, for executing 11268 queries over 101414 data. The MBT performance are independent from the number n and the value reported is the mean obtained over several running. L_{seq} is the sequential LAESA. L_{rei} is the reindexed LAESA. N_{seq} is the sequential n-Simplex. N_{rei} is the reindexed n-Simplex.*

	Cosine ($t_{0.01\%} = 0.042$)					Jensen-Shannon ($t_{0.01\%} = 0.135$)				
Dims	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*MBT*	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*MBT*
5	10.3	4.5	8.8	1.0	3.1	248.4	335.5	61.9	65.5	124.8
10	9.8	3.4	10.4	0.8		155.3	233.2	29.0	29.3	
15	12.7	2.4	11.7	**0.7**		103.5	163.2	22.3	17.2	
20	16.5	2.8	16.7	**0.7**		95.7	162.8	23.8	**14.7**	
25	17.9	2.8	17.7	0.8		87.2	155.6	25.9	16.1	
30	18.1	2.6	17.4	0.9		67.7	130.4	27.0	16.5	
35	17.7	3.1	17.1	1.1		69.6	136.3	27.9	17.2	
40	18.1	3.0	18.1	1.0		62.4	131.2	27.8	17.1	
45	17.4	2.7	18.2	1.1		61.1	133.4	29.7	18.4	
50	17.6	3.5	17.3	1.4		58.3	130.4	30.6	18.6	

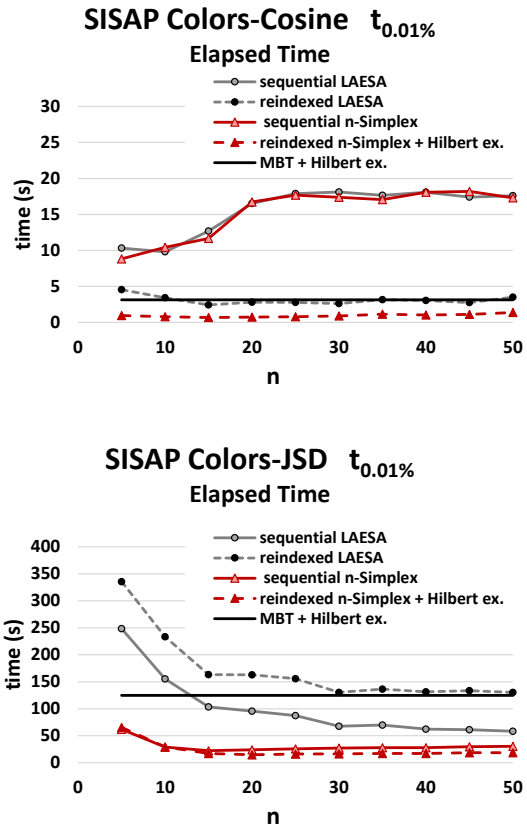


Figure 5.30: *Elapsed Times - SISAP Colors with Cosine and Jensen-Shannon distances. (Results from Table 5.6)*

Table 5.7: *SISAP* Colors*, Euclidean distance - Distance calculations performed in original and reindexed space per query. Original-space calls are the same for both sequential and reindexed mechanisms.*

	Euclidean $t_{0.01\%}$					Euclidean $t_{0.1\%}$					Euclidean $t_{1\%}$				
	Original Dist. Calls			Index Dist. Calls		Original-Distance Calls			Index Dist. Calls		Original Dist. Calls			Index Dist. Calls	
Dims	L	N	*MBT*	L_{rei}	N_{rei}	L	N	*MBT*	L_{rei}	N_{rei}	L	N	*MBT*	L_{rei}	N_{rei}
5	2747	380	1484	5275	1756	7971	2204	4628	12299	5482	23491	9267	13901	26300	16190
10	1333	54		4397	1230	4983	401		10738	4231	21748	2412		27375	13337
15	567	37		3239	1131	3278	200		9998	3777	13098	1258		21140	12094
20	510	32		3415	1150	2766	93		9600	3858	14264	1208		23774	12895
25	430	35		3149	1179	2996	86		9495	4097	10825	423		21317	12321
30	370	37		3022	1207	2359	88		8625	4055	12048	325		21932	12562
35	342	36		2848	1310	2186	59		9183	4226	8964	159		20480	12593
40	329	40		2948	1289	1891	50		7984	4097	10383	91		21506	13067
45	306	45		2822	1315	1769	48		8211	4411	8817	70		19641	13596
50	271	50		2567	1328	1680	52		7697	4319	9723	67		20996	13502

Table 5.8: *SISAP* Colors*, Cosine and Jensen-Shannon distances - Distance calculations performed in original and reindexed space per query. Original-space calls are the same for both sequential and reindexed mechanisms.*

	Cosine ($t_{0.01\%} = 0.042$					Jensen-Shannon ($t_{0.01\%} = 0.135$)				
	Original Dist. Calls			Index Dist. Calls		Original Dist. Calls			Index Dist. Calls	
Dims	L	N	*MBT*	L_{rei}	N_{rei}	L	N	*MBT*	L_{rei}	N_{rei}
5	12770	2291	5971	18400	6910	258	46	354	755	259
10	7805	578		19660	6317	116	31		649	229
15	4618	161		15458	4990	70	25		504	220
20	3892	105		15846	4804	79	25		560	209
25	3652	90		14879	4865	84	30		494	215
30	2533	79		13832	4698	73	30		457	220
35	2592	79		13559	4855	78	35		498	268
40	2142	76		13484	4635	83	40		481	266
45	1945	77		13737	4887	78	45		426	276
50	1827	78		12630	4871	89	50		443	337

the n-Simplex reach very good performance in small dimensions. For example, by 50 dimensions almost perfect accuracy is achieved for the Euclidean search with threshold $t_{0.01\%}$: on average only 50 original-space calculations per query are made to search the dataset. Since we used 50 pivots, and the distance query-pivots are used to compute the apex point, this means that the result set is not rechecked using the original distance. But in fact, even at 10 dimensions almost every apex value can be deterministically determined as either a member or otherwise of the solution set based on its upper and lower bounds. At 20 dimensions, only 10 elements of the 101414-element dataset have bounds which straddle the query threshold. This indeed reflects the fact that for $n \geq 20$ the n-simplex lower bound is practically equivalent to the Euclidean distance to search *Colors* data, as the distortion between these two distance measures is one (as shown in earlier Figure 5.27).

Equally interesting is the number of reindexed distance measurements: for n-Simplex, these are generally less than for the original space. This seems to hold for all the tested metrics and thresholds. The implication seems to be that the reindexed metric has better scalability properties than the original, although this deserves further investigation.

5.4.3.3 Upper and Lower Bound Convergence

In Section 5.4.2 we formally prove that upper and lower bounds on the original distance provided by the n-simplex projection converge to the original distance for a high value of n. Here we show an example of that convergence. We consider a scatter plot representing the relationship between the original distance and the upper and lower bounds on the n-simplex surrogate space for the various value of n in order to explore the correlation between these two quantities. For this test we consider the MIRFlickr [135] dataset (previously used in Chapters 3 and 4) for two main reasons

1. we want to inspect the case of a "real-word" dataset with possibly very high dimensional features to better appreciate the convergence of the distance bounds;

2. the MIRFlickr presents a significant number of near-duplicate images for which it would be interesting to analyse the behaviour of the distances in the surrogates space.

We used the state-of-the-art CNN features to represent the images. Specifically, as in the experiments of Section 3.2.3.1, we used the pre-trained HybridNet [276] model and we extracted the output of the first fully-connected layer (*fc6*) after applying the ReLU activation function. The resulting features are $4,096$-dimensional vectors.

The MIRFlickr dataset contains several couples of near-duplicate images each derived from the same digital source after applying some transformations, where the notion of "transformation" includes "*any operation which has been performed using a standard image editor with the intent of making cosmetic changes*" [79]. Connor et al. [79] deeply investigate the identification of the MIRFlickr near-duplicate images, and one main outcome of their research has been the publication of a set of nearly 2,000 near duplicate clusters for the MIRFlickr dataset. To discover the near-duplicate images they applied a number of different metrics to a number of different image features (based on MPEG-7 [52, 264] and Perceptual Hashing [196]) to identify candidate near-duplicates which are then validated by a human annotator. The list of the resulting $2,407$ pairs of near duplicates is publicly available [72].

We projected the CNN-features extracted from MIRFlickr into the n-simplex space for $n = 32$, 64, 128, 256, 512, 1024, 2048, 4096. We considered $5,000$ pairs of randomly selected objects and the $2,407$ pairs of near-duplicate images. For each n and for each pair of objects we plot the n-simplex lower and upper bounds as a function of the original distance. The resulting scatter plot is shown in Figure 5.31. We can observe that as n increases both the distance bounds converge to the actual distances and that the lower bound start showing a strong correlation (> 0.7) with the actual distance for $n > 500$. Moreover, as expected in such high dimensional space, the original distances between randomly selected points are all concentrated around the mean distance value. We measured the IDim [66] of the original space as $\mu^2/(2\sigma^2)$ and we found out that it is near to 310 ($\mu = 1.27$ and $\sigma = 0.051$). Interestingly, we observe that for the randomly selected points the convergence of the upper bound seems to be faster than the lower bound, while the inverse happens for the near-duplicates. We expect the latter effect since the upper bound is not a metric (identical object may have non zero "distance") and it is likely that it badly approximate the actual distance between very close objects. These facts together prompted us to

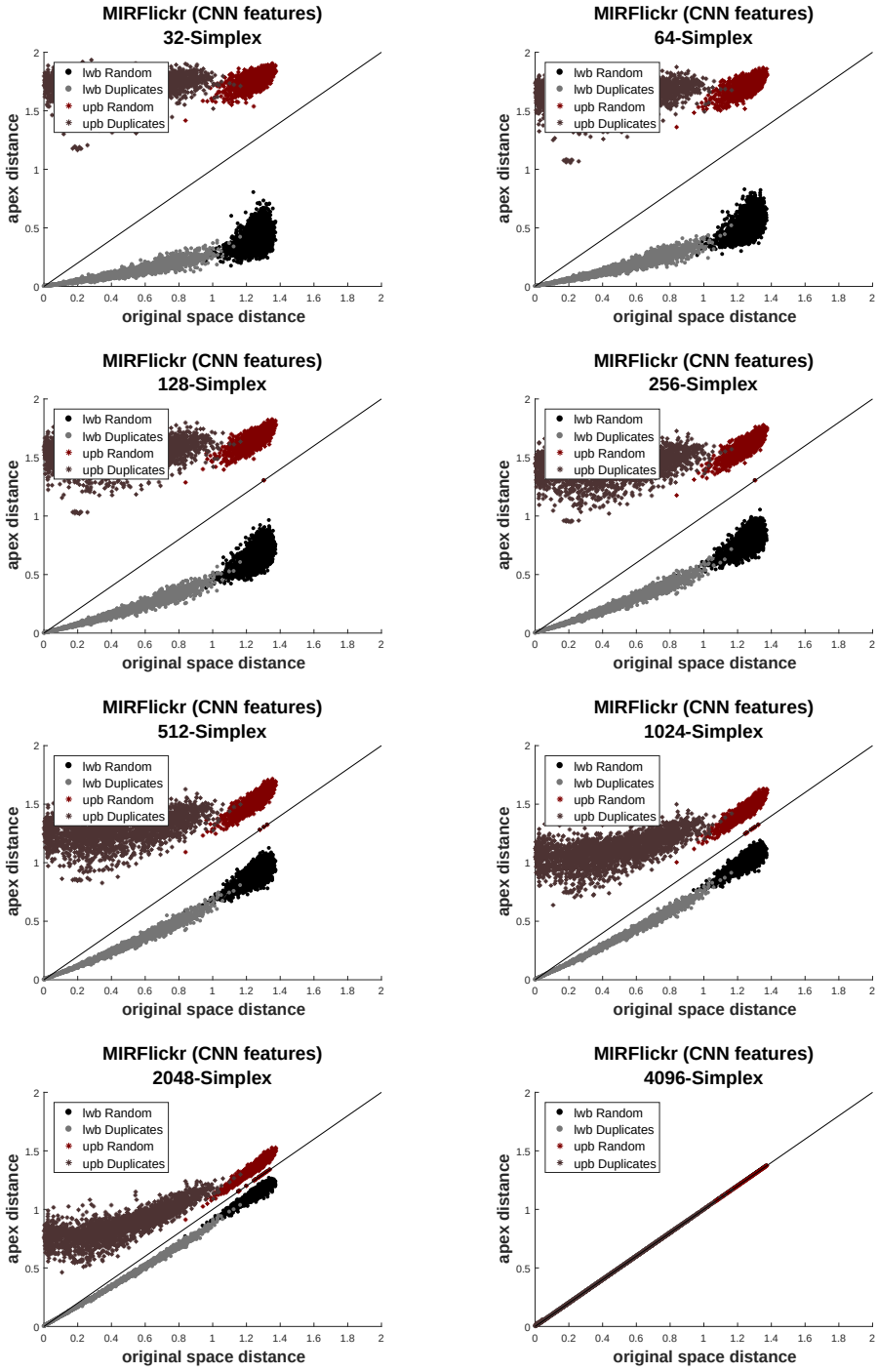


Figure 5.31: *Correlation between the original distance and either upper-bound (upb) and lower-bound (lwb) obtained in the surrogate space.*

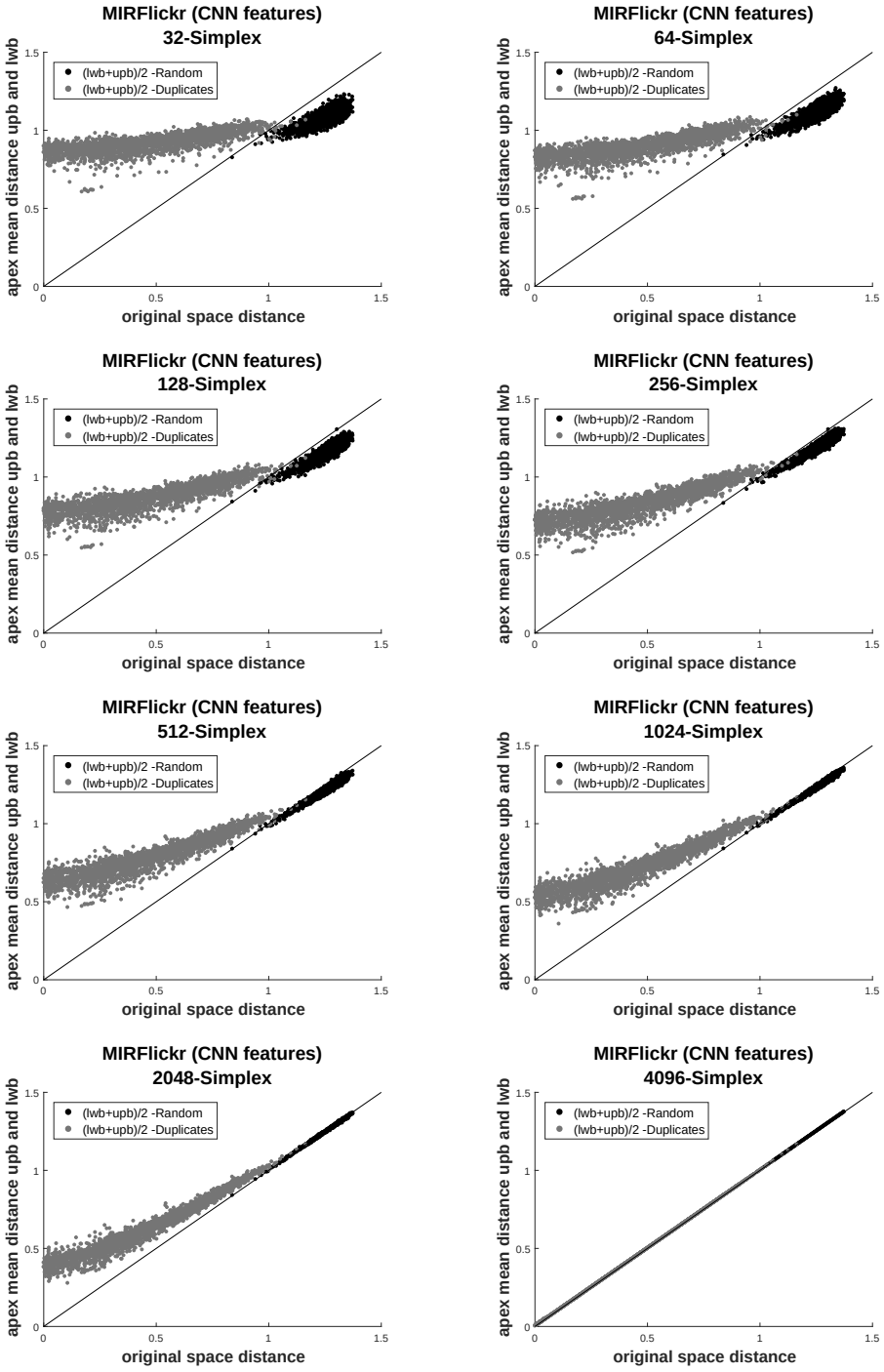


Figure 5.32: *Correlation between the original distance and the measure (UpperBound+ LowerBound)/2 obtained in the surrogate space.*

investigate the convergence of the (no metric) measure obtained as the average between the upper and the lower bounds, indicated as d_{mean} in the following. Results are shown in Figure 5.32. In the considered example, we observe that 1) the convergence of d_{mean} to the original distance is very fast for pairs of random points; 2) d_{mean} is always lower or equal to the original distance for randomly selected points; 3) d_{mean} is almost always greater or equal to the original distance for near duplicate points. Moreover, for relatively small number of n (where the upper bound badly approximate small distances) the cloud of point-distances related to the near duplicates appears almost completely separated from those related to random points. So it seems that the heuristic $d_{mean}(\phi_n(s_1), \phi_n(s_2)) > d(s_1, s_2)$, where ϕ_n is the n-simplex projection, might be used to identify very close objects, as the near duplicates. It is worth noting the near-duplicate couples used in these experiments were founded using image features completely different from the CNN feature, which makes our results more interesting.

Even if these results are preliminary we decided to report them since they might have important implication for identifying near-duplicates in domains where it is not clear which distance threshold should be used for the duplicate detection.

5.5 Summary

This Chapter was devoted to presents several theoretical and experimental results that we obtained in the context of the metric search.

We have demonstrated that many common metric spaces have geometric properties stronger than the triangle inequality: namely, the four- and n-point properties, which are expressed in term of finite isometric embeddings in Euclidean space. We have shown how these stronger geometric guarantees allow more effective metric indexing.

By using the four-point property we have derived the Hilbert exclusion condition, which significantly improves the performance of exact search for any search mechanism based on hyperplane partitioning on supermetric spaces. We believe that it is an important result in the metric search field.

We have also shown how the four-point property could, in principle, be used to construct arbitrary partition in a 2D plane into which data objects are projected, due to a lower-bound property that we have derived from the four-point property. In this respect, we have proposed the Linear Regression Tree that is a flexible data structure whose partition and exclusion conditions are tailored, at each node, to suit the distribution of the data objects.

By using the n-point property we have derived a novel projection mechanism into n-dimensional Euclidean space. We have proved that our n-simplex projection determines lower and upper bounds on unknown distances between any two data objects when the only knowledge available is their respective distances to a fixed set of reference objects within the original space. Moreover, we have proved that by increasing the number of dimensions, these bounds converge to the true distance. There are a number of ways in which the n-simplex projection can be used towards an efficient search for suitable spaces. We have so far examined only one in detail, where a Euclidean space is extracted and used to pre-filter exact search. For similarity search, the engineering tradeoffs are good: we have shown significant reductions in data size and metric cost with little loss of accuracy, leading to a significant overall improvement in exact search performance. Over the benchmark SISAP *Colors* dataset, for some different metrics, this technique gives the best-recorded performance for exact search. Finally, we have shown how our projection can be used for dimensionality reduction also in non-Euclidean space, which turns out to be very useful when dealing with very costly metrics like Jenson-Shannon and Quadratic Form distances.

CHAPTER 6

Conclusions

The recent explosion of digital visual data engenders the need of scalable Content-Based Image Retrieval systems to organize and search image collections on the basis of their visual contents. In this thesis, we have investigated and proposed efficient and effective algorithms to support various stages of a CBIR system, namely the representation of the image visual content through image features, the processing and indexing of the image features, and the image search using the query-by-example paradigm. For the latter stage, we adopt the metric search approach, which is suitable for a large number of applications and data types.

After a general introduction (Chapter 1) that points out objectives and main contributions of this thesis, in Chapter 2 we have provided the background on image representations as well as principles for indexing and searching in metric space.

Next, in Chapter 3 we have focused on efficient and effective image features for content-based search. We have started our study by addressing the question of effectiveness: we have compared state-of-the-art image features through an extensive experimental evaluation in an applicative context, which is related to recognize and retrieve objects of cultural heritage. We have shown that very high effectiveness can be achieved by combining aggregations of image local features (*e.g.* FV encoding of SIFTs) and recent CNN features. However, we have also pointed out that the boosting in the retrieval performance obtained with this features combination does not meet the need of efficient feature extraction. In fact, one main issue is that aggregations of local features have been defined and used almost exclusively with SIFT-like features, whose extraction process is costly with respect to the CNN features extraction (*e.g.* more than 1 second for the former, and less than 300ms for the latter, if using a CPU implementation). To overcome this issue we have focused on efficiency, which is of primary importance when dealing with a very large archive or when response time to a query must be very fast. In this respect, we have considered cases in which binary local features are extracted and used to describe images since their extraction process is up to two orders faster than that of non-binary features. In particular, in order to find a good trade-off between efficiency and effectiveness, we have provided a comparative analysis of aggregations of binary local features on benchmarks for image retrieval. To the best of our knowledge, such comparative analysis was missing in the research literature before our study. Moreover, we have proposed a new approach for aggregating binary vectors, named *BMM-FV*, which has proved to be more effective than other state-of-the-art aggregations of binary features. Nevertheless, we have observed that aggregation methods on binary local descriptors are more efficient but less effective than aggregation methods on

non-binary local descriptors. In order to improve the effectiveness, we have shown that the combination of CNN features and aggregations of binary features achieves high retrieval performance that is in line with that obtained by combining CNN and FV built upon the SIFTs. The advantage of the former approach is great in term of efficiency, due to the faster feature extraction.

In Chapter 4 we have proposed the *Blockwise Surrogate Text Representation (BSTR)* and the *Deep Permutations*, which are two promising approaches for processing some state-of-the-art image descriptors (*e.g.* VLAD and CNN features) in order to efficiently/effectively index them. In particular, the BSTR encodes compound metric objects into textual representations, which allows similarity search to be performed using off-the-shelf text search engine. We have shown that our approach is notably effective to index VLAD image features, for which we obtain a retrieval performance on a benchmark dataset that is even better than that obtained using the original VLAD vectors. Also the results in term of response times are good; however, one drawback of our technique is the memory occupation which is higher than that required by similar approaches, such as STR [109]. The Deep Permutations, instead, revealed to be a very efficient and effective technique for encoding deep features into permutations to be subsequently indexed using any permutation-based approach. Each "deep permutation" is computed by ordering the vector values of a deep feature rather than measuring the distances between the feature and a set of pivots (as traditionally done in permutation-based approach). Therefore, our approach results to be a very efficient way for building a permutation representation. Moreover, we experimentally proved that our Deep Permutations achieves very high effectiveness.

Finally, in Section 5 we have focused on similarity search on metric space. In particular, we have investigated a class of metric spaces, which we have called supermetrics, that have geometric guaranties stronger than that derived by the triangle inequality. For those spaces, we have presented a number of theoretical and experimental results related to metric indexing and searching. We have observed that many properties and distance bounds defined for metric spaces can be re-read in the light of finite isometric embeddings into Euclidean spaces. Using the four-point property we have derived the *Hilbert exclusion* condition that we have proved to be weaker than the hyperbolic exclusion, which is actually used in quite all the metric indexes based on hyperplane partitioning. The weakness of the Hilbert exclusion guarantees more effective pruning of the search space, and so lead to save distance computations at query time. Therefore, the efficiency of every index that uses the hyperbolic exclusion can be notably increased by using our Hilbert exclusion. Moreover, we have shown how the four-point property can be used to project all the points of a supermetric space into a 2D Euclidean plane, where the distance between the projected points is a lower bounding of the original distance. We have referred this embedding to as *tetrahedral projection* and we have shown that this allows defining novel partitioning and indexing approaches. One drawback of our techniques is that they can be applied only to supermetric spaces. However, we have proved that many common metric spaces are supermetrics, as for example spaces of any dimensions with the Euclidean, Jenson-Shannon, Cosine and Quadratic Form distances. We have also shown that most of the supermetrics used in applications have the stronger n-point property, which allowed us to investigates a novel projection procedure that generalizes the tetrahedral projection. Specifically, we have defined the *n-Simplex projection* that embeds a space with the n-point property into an n-dimensional Euclidean space by using as knowledge only the distances between data objects and a set of n pivots. We have proved that the n-Simplex projection provides arbitrarily tight lower and upper bounds on distances between data objects. This is particularly useful when dealing with very costly metrics like Jenson-Shannon and Quadratic Form distances; in facts, our projections allows working in a space governed by the cheaper Euclidean distance. Finally, we have experimentally shown that our n-Simplex projection can be profitably used for dimensionality reduction (also in non-Euclidean space) as well as for indexing and searching tasks.

6.1 Future Work

In this thesis, we have developed foundations for supermetric search and proposed several techniques to support content-based image search, however, there is a number of theoretical and applicative aspects that can be further explored. The most promising ones are presented below, organized by the relevant topic.

Aggregations of binary features. Recent works based on CNNs suggest that techniques for aggregating binary local features (see Section 3) could be further applied to deep features. In fact, on one hand, local features based on CNNs, aggregated with traditional VLAD and FV approaches, have been proposed to obtain robustness to geometric deformations [254, 269]. On the other hand, binarizations of global CNN features have been also proposed in [157, 168]. So a future research direction may be applying our BMM-FV approach over binary deep local descriptors leveraging on the local and binary approaches mentioned above. Moreover, we note that our BMM-FV is a general technique for encoding a set of binary vectors into a single descriptor by computing statistical summaries of the given data. Therefore, we believe that there are other applicative contexts that could further exploit our encoding schema.

Deep Permutations. In Section 4.2 we proposed the Deep Permutations to encode deep features, but we tested it only on CNN features. A future work is investigating the use of this approach to represent and index other kinds of deep features, as partially done in [119] and [22] that recently applied our technique to represent and index RNN features [112] and R-MAC features [113,247], respectively. Moreover, different approaches to handle the presence of negative or zero-values in the vectors could be further investigated as alternative to our "zeros-to-l" and "noReLu" approaches (Section 4.2). Finally, we observe that the idea of creating the permutation by ordering the vector values could generalize to other multidimensional features. We believe this is an input for future research investigations.

Supermetric search and n-point property. The study presented in Chapter 5 is the first step towards enhancing our understanding of similarity search on *supermetric* space. However, many issues worthy of further investigation.

In [128, 172] the family of Ptolemaic metrics, *i.e.* metrics that satisfy the Ptolemy's inequality $(d(x,v)d(y,u) \leq d(x,y)d(u,v) + d(x,u)d(y,v)$ for all objects $x, y, u, v)$ were studied in the context of metric search. In particular, it has been shown that the Ptolemy's inequality allows deriving distance bounds with filtering power higher than that obtained using the triangle inequality. We now observe that every supermetric space is also a Ptolemaic space, but the distance bounds obtained by our approaches are different from that obtained with the Ptolemy's inequality. These observations raise questions in need of further investigation: Are the supermetric spaces or the spaces with n-point property equivalent to Ptolemaic spaces? Is it possible to derive mechanisms using both the properties (Ptolemy's inequality and four-point property) to further increase the efficiency of supermetric indexing? Conversely, which property does provide tighter distance bounds? These topics and related issues are deferred to future work.

One limitation of our study is that it treats only the exact search scenario using range queries. We are currently in the process of investigating the use of the four-point property and the n-Simplex projection to performs approximate similarity search, also using k-NN search queries. We also observe that the distance bounds obtained with the tetrahedral projection, or more generally with the n-Simplex projection, might be integrated into existing approximate search algorithms. To this respect, we are planning to use them on the MI-File [36].

An additional interesting aspect that we have left for future work is analysing the impact of the choice of the reference objects in the n-Simplex projection with the aim of minimizing the distortion of the distance bounds for a fixed value of n. In facts, several pivot selection techniques (*e.g.* k-means [171], k-medoids [151], BPP [24], FFT [111], and PSIS [60]) could be tested in conjunction with our approach, or even a new pivot selection technique could be defined in order to optimize our filtering conditions.

Our experiments on Section 5.4.3.3 suggest that the distance bounds obtained with the n-Simplex projection could be a useful aid for the problem of the near-duplicates detection. We plan to further inspect this conjecture.

Finally, we believe that our tetrahedral and 3-Simplex projections could be further exploited for visualization tasks in 2D and 3D spaces, respectively. In particular, different projections could be used to visualize different portions (or clusters) of the data, also choosing the reference points used to projects the objects in a dynamic way in order to "visually navigate" a given dataset.

APPENDIX $\mathcal{A}$

BMM-FV Computation

In the following we explicitly derive the score vector and the approximation of the Fisher Information Matrix used to define our BMM-FV (Section 3.2.2). Throughout this appendix we have used $[\![\cdot]\!]$ notation to represent the Iverson bracket which equals one if the arguments is true, and zero otherwise.

A.1 Score vector computation

In the following, we have reported the computation of the score function $\boldsymbol{G}_{\lambda}^{X}$, defined as the gradient of the log-likelihood of a data X with respect to the parameters λ of the generative model $p(\cdot|\lambda)$, in the case where a Bernoulli Mixture Model is used. We considered a multivariate Bernoulli mixture with K components and parameters $\lambda = \{w_k, \mu_{kd},\, k = 1, \ldots, K,\, d = 1, \ldots, D\}$:

$$p(\boldsymbol{x}_t|\lambda) = \sum_{k=1}^{K} w_k p_k(x_t) \tag{A.1}$$

where

$$p_k(\boldsymbol{x}_t) = \prod_{d=1}^{D} \mu_{kd}^{x_{td}} (1 - \mu_{kd})^{1 - x_{td}} \tag{A.2}$$

and

$$\sum_{k=1}^{K} w_k = 1, \quad w_k > 0 \qquad \forall\, k = 1, \ldots, K. \tag{A.3}$$

Under the independence assumption, the Fisher score with respect to the generic parameter λ_k is expressed as: $G_{\lambda_k}^{X} = \sum_{t=1}^{T} \dfrac{\partial \log p(\boldsymbol{x}_t|\lambda)}{\partial \lambda_k} = \sum_{t=1}^{T} \dfrac{1}{p(\boldsymbol{x}_t|\lambda)} \dfrac{\partial}{\partial \lambda_k} \left[\sum_{i=1}^{K} w_i p_i(\boldsymbol{x}_t) \right]$.

To avoid enforcing explicitly the constraints in (A.3), we used the soft-max formalism [154, 226] for the weight parameters: $w_k = \exp(\alpha_k) / \sum_{i=1}^{K} \exp(\alpha_i)$.

To compute $\frac{\partial}{\partial \lambda_k}\left[\sum_{i=1}^{K} w_i p_i(\boldsymbol{x}_t)\right]$, we first observe that

$$\begin{aligned}\frac{\partial w_i}{\partial \alpha_k} &= \frac{\partial}{\partial \alpha_k}\left[\frac{\exp(\alpha_i)}{\sum_{j=1}^{K}\exp(\alpha_j)}\right] \\ &= \frac{\exp(\alpha_k)\left(\sum_{j=1}^{K}\exp(\alpha_j)\right)[\![i=k]\!] - \exp(\alpha_i)\exp(\alpha_k)}{\left(\sum_{j=1}^{K}\exp(\alpha_j)\right)^2} \\ &= w_k[\![i=k]\!] - w_k w_i\end{aligned} \tag{A.4}$$

and

$$\begin{aligned}\frac{\partial p_i(\boldsymbol{x}_t)}{\partial \mu_{kd}} &= \frac{\partial}{\partial \mu_{kd}}\left[\prod_{l=1}^{D}\mu_{kl}^{x_{tl}}\left(1-\mu_{kl}\right)^{1-x_{tl}}\right][\![i=k]\!] \\ &= \left([\![x_{td}=1]\!]-[\![x_{td}=0]\!]\right)\left(\prod_{\substack{l=1\\ l\neq d}}^{D}\mu_{kl}^{x_{tl}}\left(1-\mu_{kl}\right)^{1-x_{tl}}\right)[\![i=k]\!] \\ &= \left([\![x_{td}=1]\!]-[\![x_{td}=0]\!]\right)\left(\frac{p_k(\boldsymbol{x}_t)}{\mu_{kd}^{x_{td}}\left(1-\mu_{kd}\right)^{1-x_{td}}}\right)[\![i=k]\!] \\ &= p_k(\boldsymbol{x}_t)\left(\frac{(1-\mu_{kd})[\![x_{td}=1]\!]-\mu_{kd}[\![x_{td}=0]\!]}{\mu_{kd}(1-\mu_{kd})}\right)[\![i=k]\!] \\ &= p_k(\boldsymbol{x}_t)\left(\frac{x_{td}-\mu_{kd}}{\mu_{kd}(1-\mu_{kd})}\right)[\![i=k]\!].\end{aligned} \tag{A.5}$$

Hence, the Fisher score with respect to the parameter α_k is obtained as

$$\begin{aligned}G_{\alpha_k}^{X} &= \sum_{t=1}^{T}\sum_{i=1}^{K}\frac{p_i(\boldsymbol{x}_t)}{p(\boldsymbol{x}_t|\lambda)}\frac{\partial w_i}{\partial \alpha_k} \overset{(A.4)}{=} \sum_{t=1}^{T}\sum_{i=1}^{K}\frac{p_i(\boldsymbol{x}_t)}{p(\boldsymbol{x}_t|\lambda)}w_k\left([\![i=k]\!]-w_i\right) \\ &= \sum_{t=1}^{T}\left(\frac{p_k(\boldsymbol{x}_t)}{p(\boldsymbol{x}_t|\lambda)}w_k - \sum_{i=1}^{K}\frac{p_i(\boldsymbol{x}_t)}{p(\boldsymbol{x}_t|\lambda)}w_k w_i\right) = \sum_{t=1}^{T}\left(\gamma_t(k) - w_k\sum_{i=1}^{K}\gamma_t(i)\right) \\ &= \sum_{t=1}^{T}\left(\gamma_t(k)-w_k\right)\end{aligned} \tag{A.6}$$

and the Fisher score related to the parameter μ_{kd} is

$$\begin{aligned}G_{\mu_{kd}}^{X} &= \sum_{t=1}^{T}\frac{\partial \log p(\boldsymbol{x}_t|\lambda)}{\partial \mu_{kd}} = \sum_{t=1}^{T}\frac{1}{p(\boldsymbol{x}_t|\lambda)}\frac{\partial}{\partial \mu_{kd}}\left[\sum_{i=1}^{K}w_i p_i(\boldsymbol{x}_t)\right] \\ &= \sum_{t=1}^{T}\frac{w_k}{p(\boldsymbol{x}_t|\lambda)}\frac{\partial p_k(\boldsymbol{x}_t)}{\partial \mu_{kd}} \overset{(A.5)}{=} \sum_{t=1}^{T}\frac{w_k p_k(\boldsymbol{x}_t)}{p(\boldsymbol{x}_t|\lambda)}\left(\frac{x_{td}-\mu_{kd}}{\mu_{kd}(1-\mu_{kd})}\right) \\ &= \sum_{t=1}^{T}\gamma_t(k)\left(\frac{x_{td}-\mu_{kd}}{\mu_{kd}(1-\mu_{kd})}\right).\end{aligned} \tag{A.7}$$

A.2 Approximation of the Fisher Information Matrix

Our derivation of the FIM is based on the assumption (see also [208, 226]) that for each observation $\boldsymbol{x} = [x_1, \cdots, x_D] \in \{0,1\}^D$ the distribution of the occupancy probability $\gamma(\cdot) = p(\cdot|\boldsymbol{x}, \lambda)$ is sharply peaking, i.e. there is one Bernoulli index k such that $\gamma_x(k) \approx 1$ and $\forall\, i \neq k$, $\gamma_x(i) \approx 0$. This assumption implies that

$$\begin{aligned}&\gamma_x(k)\gamma_x(i) \approx 0 \quad \forall\, k, i = 1\ldots, K, i \neq k \\ &\gamma_x(k)^2 \approx \gamma_x(k) \quad \forall\, k = 1, \ldots, K\end{aligned}$$

and then

$$\gamma_x(k)\gamma_x(i) \approx \gamma_x(k)[\![i=k]\!], \tag{A.8}$$

where $[\![\cdot]\!]$ is the Iverson bracket.
The elements of the FIM are defined as:

$$[\boldsymbol{F}_\lambda]_{i,j} = \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\left(\frac{\partial \log p(x|\lambda)}{\partial \lambda_i}\right)\left(\frac{\partial \log p(x|\lambda)}{\partial \lambda_j}\right)\right]. \tag{A.9}$$

Hence, the FIM $\boldsymbol{F}_\lambda$ is symmetric and can be written as block matrix

$$\boldsymbol{F}_\lambda = \begin{bmatrix} \boldsymbol{F}_{\alpha,\alpha} & \boldsymbol{F}_{\mu,\alpha} \\ \boldsymbol{F}_{\mu,\alpha}^\top & \boldsymbol{F}_{\mu,\mu} \end{bmatrix}.$$

By using the definition of the occupancy probability (i.e. $\gamma_x(k) = w_k p_k(\boldsymbol{x})/p(\boldsymbol{x}|\lambda)$) and the fact that p_k is the distribution of a D-dimensional Bernoulli of mean $\boldsymbol{\mu}_k$, we have the following useful equalities:

$$\mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\gamma_x(k)\right] = \sum_{\boldsymbol{x}\in\{0,1\}^D} \gamma_x(k)p(\boldsymbol{x}|\lambda) = w_k \tag{A.10}$$

$$\mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\gamma_x(k)x_d\right] = w_k\mu_{kd} \tag{A.11}$$

$$\mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\gamma_x(k)x_d x_l\right] = w_k\mu_{kd}\left(\mu_{kl}[\![d\neq l]\!] + [\![d=l]\!]\right) \tag{A.12}$$

$$\mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\frac{\partial \log p(x|\lambda)}{\partial \alpha_k}\right] \overset{(A.6)}{=} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\gamma_x(k) - w_k\right] = 0 \tag{A.13}$$

$$\mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\frac{\partial \log p(x|\lambda)}{\partial \mu_{id}}\right] \overset{(A.7)}{=} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\frac{\gamma_x(k)(x_d-\mu_{kd})}{\mu_{kd}(1-\mu_{kd})}\right] = 0. \tag{A.14}$$

It follows that $\boldsymbol{F}_\lambda$ may approximated by a diagonal block matrix, because the mixing blocks $\boldsymbol{F}_{\mu_{kd},\alpha_i}$ are close to the zero matrix:

$$\begin{aligned} F_{\mu_{kd},\alpha_i} &= \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\left(\frac{\partial \log p(x|\lambda)}{\partial \mu_{kd}}\right)\left(\frac{\partial \log p(x|\lambda)}{\partial \alpha_i}\right)\right] \\ &\overset{(A.6)-(A.7)}{=} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\gamma_x(k)\frac{(x_d-\mu_{kd})}{\mu_{kd}(1-\mu_{kd})}(\gamma_x(i)-w_i)\right] \\ &\overset{(A.8)}{\approx} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\frac{\gamma_x(k)(x_d-\mu_{kd})}{\mu_{kd}(1-\mu_{kd})}\right]([\![i=k]\!]-w_i) \\ &\overset{(A.14)}{=} 0. \end{aligned}$$

The block $\boldsymbol{F}_{\mu,\mu}$ can be written as $KD\times KD$ diagonal matrix, in fact:

$$\begin{aligned} F_{\mu_{id},\mu_{kl}} &\overset{(A.9)}{=} \mathbb{E}\left[\left(\frac{\partial \log p(x|\lambda)}{\partial \mu_{id}}\right)\left(\frac{\partial \log p(x|\lambda)}{\partial \mu_{kl}}\right)\right] \\ &\overset{(A.7)}{=} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\gamma_x(i)\gamma_x(k)\frac{(x_d-\mu_{id})}{\mu_{id}(1-\mu_{id})}\frac{(x_l-\mu_{kl})}{\mu_{kl}(1-\mu_{kl})}\right] \\ &\overset{(A.8)}{\approx} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\frac{\gamma_x(k)(x_d-\mu_{kd})(x_l-\mu_{kl})}{\mu_{kd}\mu_{kl}(1-\mu_{kd})(1-\mu_{kl})}\right][\![i=k]\!] \\ &\overset{(A.10)-(A.12)}{=} \frac{w_k(\mu_{kd}\mu_{kl}[\![d\neq l]\!]+\mu_{kl}[\![d=l]\!]-\mu_{kd}\mu_{kl})}{\mu_{kd}\mu_{kl}(1-\mu_{kd})(1-\mu_{kl})}[\![i=k]\!] \\ &= \frac{w_k(\mu_{kd}[\![d\neq l]\!]+[\![d=l]\!]-\mu_{kd})}{\mu_{kd}(1-\mu_{kd})(1-\mu_{kl})}[\![i=k]\!] \\ &= \frac{w_k}{\mu_{kd}(1-\mu_{kd})}[\![i=k]\!][\![d=l]\!]. \end{aligned} \tag{A.15}$$

The relation (A.15) points that the diagonal elements of our FIM approximation are $w_k/\mu_{kd}(1-\mu_{kd})$ and the corresponding entries in $\boldsymbol{L}_\lambda$ (i.e. the square root of the inverse of FIM) equal $\sqrt{\mu_{kd}(1-\mu_{kd})/w_k}$.

Appendix A. BMM-FV Computation

The block related to the $\boldsymbol{\alpha}$ parameters is $\boldsymbol{F}_{\alpha,\alpha} = (\text{diag}(\boldsymbol{w}) - \boldsymbol{w}\boldsymbol{w}^\top)$ where $\boldsymbol{w} = [w_1, \cdots, w_K]$, in fact:

$$\begin{aligned} F_{\alpha_k,\alpha_i} &\overset{(A.9)}{=} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[\left(\frac{\partial \log p(x|\lambda)}{\partial \alpha_k}\right)\left(\frac{\partial \log p(x|\lambda)}{\partial \alpha_i}\right)\right] \\ &\overset{(A.6)}{=} \mathbb{E}_{\boldsymbol{x}\sim p(\cdot|\lambda)}\left[(\gamma_x(k) - w_k)(\gamma_x(i) - w_i)\right] \\ &\overset{(A.8)}{\approx} \mathbb{E}_{p(\cdot|\lambda)}\left[\gamma_x(k)[\![i=k]\!] - \gamma_x(k)w_i - \gamma_x(i)w_k + w_i w_k\right] \\ &\overset{(A.10)-(A.11)}{=} (w_k[\![i=k]\!] - w_i w_k)\,. \end{aligned}$$

The matrix $\boldsymbol{F}_{\alpha,\alpha}$ is not invertible (indeed $\boldsymbol{F}_{\alpha,\alpha}[1,\cdots,1] = \boldsymbol{0}$) due to the dependence of the mixing weights ($\sum_{i=1}^K \alpha_i = \sum_{i=1}^K w_i = 1$). Since there are only $K-1$ degrees of freedom in the mixing weight, as proposed in [226], we can fix α_K equal to a constant without loss of generality and work with a reduced set of $K-1$ parameters: $\tilde{\boldsymbol{\alpha}} = [\alpha_1, \ldots, \alpha_{K-1}]$.

Taking into account the Fisher score with respect to $\tilde{\alpha}$, i.e.

$$\boldsymbol{G}^X_{\tilde{\alpha}} = \boldsymbol{\nabla}_{\tilde{\alpha}} \log p(X|\lambda) = [G^X_{\alpha_1}, \ldots, G^X_{\alpha_{K-1}}] = \widetilde{\boldsymbol{G}^X_{\alpha}},$$

the corresponding block of the FIM is $\boldsymbol{F}_{\tilde{\alpha},\tilde{\alpha}} = (\text{diag}(\tilde{\boldsymbol{w}}) - \tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^\top)$, where $\tilde{\boldsymbol{w}} = [w_1, \ldots, w_{K-1}]$. The matrix $\boldsymbol{F}_{\tilde{\alpha},\tilde{\alpha}}$ is invertible, indeed it can be decomposed into a product of an invertible diagonal matrix $\boldsymbol{D} = \text{diag}(\tilde{\boldsymbol{w}})$ and an invertible elementary matrix [1] $\boldsymbol{E}(\mathbf{e}, \tilde{\boldsymbol{w}}, -1) = \boldsymbol{I} - \mathbf{e}\tilde{\boldsymbol{w}}^\top$, where $\mathbf{e} = [1, \cdots, 1]$; its inverse is

$$\boldsymbol{F}^{-1}_{\tilde{\alpha},\tilde{\alpha}} = \text{diag}(\tilde{\boldsymbol{w}})^{-1}\left(\boldsymbol{I} + \frac{1}{\sum_{i=1}^{K-1} w_i - 1}\mathbf{e}\tilde{\boldsymbol{w}}^\top\right) = \left(\text{diag}(\tilde{\boldsymbol{w}})^{-1} + \frac{1}{w_K}\mathbf{e}\mathbf{e}^\top\right).$$

It follows that

$$K_{\tilde{\alpha}}(X,Y) = (\boldsymbol{G}^X_{\tilde{\alpha}})^\top \boldsymbol{F}^{-1}_{\tilde{\alpha},\tilde{\alpha}} \boldsymbol{G}^Y_{\tilde{\alpha}} = \left((\boldsymbol{G}^X_{\tilde{\alpha}})^\top \text{diag}(\tilde{\boldsymbol{w}})^{-1} \boldsymbol{G}^Y_{\tilde{\alpha}} + \frac{1}{w_K}(\mathbf{e}^\top \boldsymbol{G}^X_{\tilde{\alpha}})(\mathbf{e}^\top \boldsymbol{G}^Y_{\tilde{\alpha}})\right) = \sum_{k=1}^K \frac{\boldsymbol{G}^X_{\alpha_k}\boldsymbol{G}^Y_{\alpha_k}}{w_k}$$

where we used $\mathbf{e}^\top \boldsymbol{G}^Z_{\tilde{\alpha}} = \sum_{k=1}^{K-1}\sum_{z\in Z}(\gamma_z(k) - w_k) = -\sum_{z\in Z}(\gamma_z(K) - w_K) = -G^Z_{\alpha_K}$.
By defining $\mathcal{G}^X_{\alpha_k} = \frac{1}{\sqrt{w_k}}\sum_{x\in X}(\gamma_x(k) - w_k)$, we finally obtain $K_{\tilde{\alpha}}(X,Y) = \left(\boldsymbol{\mathcal{G}}^X_{\alpha}\right)^\top \boldsymbol{\mathcal{G}}^Y_{\alpha}$. Please note that we don't need to explicitly compute the Cholesky decomposition of the matrix $\boldsymbol{F}^{-1}_{\tilde{\alpha},\tilde{\alpha}}$ because the Fisher Kernel $K_{\tilde{\alpha}}(X,Y)$ can be easily rewritten as dot product between the feature vector $\boldsymbol{\mathcal{G}}^X_{\alpha}$ and $\boldsymbol{\mathcal{G}}^Y_{\alpha}$.

[1] An elementary matrix $\boldsymbol{E}(\boldsymbol{u},\boldsymbol{v},\sigma) = \boldsymbol{I} - \sigma\boldsymbol{u}\boldsymbol{v}^H$ is non-singular if and only if $\sigma\boldsymbol{v}^H\boldsymbol{u} \neq 1$ and in this case the inverse is $\boldsymbol{E}(\boldsymbol{u},\boldsymbol{v},\sigma)^{-1} = \boldsymbol{E}(\boldsymbol{u},\boldsymbol{v},\tau)$ where $\tau = \sigma/(\sigma\boldsymbol{v}^H\boldsymbol{u} - 1)$. More details on this topic can be found in [134].

Bibliography

[1] Amazon Mobile. http://www.amazon.com/. Last visit: March, 2018.

[2] Bing Images. http://www.bing.com/images/. Last visit: March, 2018.

[3] Epigraphic Database Rome. http://www.edr-edr.it/. Last visit: March, 2018.

[4] Google Googles. http://www.google.com/mobile/goggles/. Last visit: March, 2018.

[5] Google Images. https://images.google.com/. Last visit: March, 2018.

[6] Metric Space Framework. https://bitbucket.org/richardconnor/metric-space-framework.git/. Last visit: March, 2018.

[7] Museum Mobile Guide - Pinacoteca di Brera. https://mmg.inera.it/frontend/pinacoteca-di-brera-2 . Last visit: March, 2018.

[8] Nasa Dataset. http://www.dimacs.rutgers.edu/Challenges/Sixth/software.html . Last visit: March, 2018.

[9] Paris6k Dataset. http://www.robots.ox.ac.uk/ vgg/data/parisbuildings/. Last visit: March, 2018.

[10] Pisa Dataset. http://www.nmis.isti.cnr.it/falchi/pisaDataset/. Last visit: March, 2018.

[11] Statistic Brain - Instagram Company Statistics. http://www.statisticbrain.com/instagram-company-statistics/. Last visit: March, 2018.

[12] The EAGLE project. http://www.eagle-network.eu/. Last visit: March, 2018.

[13] ViSenze. https://www.visenze.com/. Last visit: March, 2018.

[14] Yandex. https://yandex.com/images/. Last visit: March, 2018.

[15] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. FREAK: Fast Retina Keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517, June 2012.

[16] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. *KAZE Features*, pages 214–227. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[17] Pablo Fernández Alcantarilla, Jesús Nuevo, and Adrien Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *In British Machine Vision Conference (BMVC)*, 2013.

[18] Giuseppe Amato, Paolo Bolettieri, Falchi Fabrizio, and Vadicamo Lucia. Sistema di riconoscimento delle immagini e mobile app. *Forma Vrbis*, 11:22–25, 2016.

[19] Giuseppe Amato, Paolo Bolettieri, Fabrizio Falchi, and Claudio Gennaro. Large scale image retrieval using vector of locally aggregated descriptors. In Nieves Brisaboa, Oscar Pedreira, and Pavel Zezula, editors, *Similarity Search and Applications*, volume 8199 of *Lecture Notes in Computer Science*, pages 245–256. Springer Berlin Heidelberg, 2013.

[20] Giuseppe Amato, Paolo Bolettieri, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. Using Apache Lucene to Search Vector of Locally Aggregated Descriptors. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016)*, pages 383–392, 2016.

[21] Giuseppe Amato, Paolo Bolettieri, Fabrizio Falchi, Fausto Rabitti, and Lucia Vadicamo. Visual recognition in the EAGLE project. In *CEUR Workshop Proceedings of the 6th Italian Information Retrieval Workshop, (IIR 2015)*, volume 1404, 2015.

[22] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. Efficient indexing of regional maximum activations of convolutions using full-text search engines. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 420–423. ACM, 2017.

[23] Giuseppe Amato, Franca Debole, Fabrizio Falchi, Claudio Gennaro, and Fausto Rabitti. Large scale indexing and searching deep convolutional neural network features. In *Proceeding of the 18th International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2016)*. Springer, 2016. to appear.

[24] Giuseppe Amato, Andrea Esuli, and Fabrizio Falchi. A comparison of pivot selection techniques for permutation-based indexing. *Information Systems*, 52:176 – 188, 2015. Special Issue on Selected Papers from {SISAP} 2013.

[25] Giuseppe Amato, Fabrizio Falchi, and Claudio Gennaro. On reducing the number of visual words in the bag-of-features representation. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISIGRAPP 2013)*, pages 657–662, 2013.

[26] Giuseppe Amato, Fabrizio Falchi, and Claudio Gennaro. Fast image classification for monument recognition. *J. Comput. Cult. Herit.*, 8(4):18:1–18:25, August 2015.

[27] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Paolo Bolettieri. Indexing vectors of locally aggregated descriptors using inverted files. In *Proceedings of International Conference on Multimedia Retrieval*, ICMR '14, pages 439:439–439:442, 2014.

[28] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Fausto Rabitti. *YFCC100M-HNfc6: A Large-Scale Deep Features Benchmark for Similarity Search*, pages 196–209. Springer International Publishing, Cham, 2016.

[29] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. *Deep Permutations: Deep Convolutional Neural Networks and Permutation-Based Indexing*, pages 93–106. Springer International Publishing, 2016.

[30] Giuseppe Amato, Fabrizio Falchi, Fausto Rabitti, and Lucia Vadicamo. Inscriptions visual recognition. A comparison of state-of-the-art object recognition approaches. In *Proceedings of the First EAGLE International Conference*, volume 26, pages 117 –131. Sapienza Universitá Editrice, 2014.

[31] Giuseppe Amato, Fabrizio Falchi, Fausto Rabitti, and Lucia Vadicamo. *Some Theoretical and Experimental Observations on Permutation Spaces and Similarity Search*, pages 37–49. Springer International Publishing, Cham, 2014.

[32] Giuseppe Amato, Fabrizio Falchi, Fausto Rabitti, and Lucia Vadicamo. Combining Fisher Vector and Convolutional Neural Networks for image retrieval. In *CEUR Workshop Proceedings of the 7th Italian Information Retrieval Workshop, (IIR 2016)*, volume 1653, 2016.

[33] Giuseppe Amato, Fabrizio Falchi, and Lucia Vadicamo. How effective are aggregation methods on binary features? In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, (VISIGRAPP 2016)*, pages 566–573, 2016.

[34] Giuseppe Amato, Fabrizio Falchi, and Lucia Vadicamo. Visual recognition of ancient inscriptions Using Convolutional Neural Network and Fisher Vector. *Journal on Computing and Cultural Heritage*, 9(4):21:1–21:24, December 2016.

[35] Giuseppe Amato, Fabrizio Falchi, and Lucia Vadicamo. Aggregating binary local descriptors for image retrieval. *Multimedia Tools and Applications*, pages 1–31, 2017.

[36] Giuseppe Amato, Claudio Gennaro, and Pasquale Savino. MI-File: Using inverted files for scalable approximate similarity search. *Multimedia Tools and Applications- An International Journal*, (Online first), November 2012 2012.

[37] Giuseppe Amato, Andrea Mannocci, Lucia Vadicamo, and Franco Zoppi. Coping with interoperability in cultural heritage data infrastructures: the europeana network of ancient greek and latin epigraphy. In *Proccedings of the 6th AIUCD Conference: "Il telescopio inverso: big data e distant reading nelle discipline umanistiche" ()AIUCD 2017)*, 2017.

[38] Giuseppe Amato and Pasquale Savino. Approximate similarity search in metric spaces using inverted files. In *Proceedings of the 3rd international conference on Scalable information systems*, InfoScale '08, pages 28:1–28:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[39] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[40] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918, June 2012.

[41] Relja Arandjelović and Andrew Zisserman. All about VLAD. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1578–1585, June 2013.

[42] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991.

[43] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *Computer Vision–ECCV 2014*, pages 584–599. Springer, 2014.

[44] Ricardo A. Baeza-Yates, Walter Cunto, Udi Manber, and Sun Wu. Proximity matching using fixed-queries trees. In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, CPM '94, pages 198–212, London, UK, UK, 1994. Springer-Verlag.

[45] Juan Manuel Barrios, Benjamin Bustos, and Tomáš Skopal. Analyzing and dynamically indexing the query set. *Information Systems*, 45:37–47, 2014.

[46] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. Lsh forest: Self-tuning indexes for similarity search. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 651–660, New York, NY, USA, 2005. ACM.

[47] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, 2006.

[48] Paul R Beaudet. Rotationally invariant image operators. In *Proceedings of the 4th International Joint Conference on Pattern Recognition*, pages 579–583, Kyoto, Japan, November 1978.

[49] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.

[50] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[51] Leonard M. Blumenthal. *Theory and applications of distance geometry*. Clarendon Press, 1953.

[52] Miroslaw Bober. Mpeg-7 visual shape descriptors. *IEEE Transactions on circuits and systems for video technology*, 11(6):716–719, 2001.

[53] Paolo Bolettieri, Vittore Casarosa, Fabrizio Falchi, Lucia Vadicamo, Philippe Martineau, Silvia Orlandi, and Raffaella Santucci. *Searching the EAGLE Epigraphic Material Through Image Recognition via a Mobile Device*, pages 351–354. Springer International Publishing, Cham, 2015.

[54] Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, Tommaso Piccioli, and Fausto Rabitti. CoPhIR: a test collection for content-based image retrieval. *CoRR*, abs/0905.4627v2, 2009. http://cophir.isti.cnr.it.

[55] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2559–2566, June 2010.

[56] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 2000.

[57] Sergey Brin. Near neighbor search in large metric spaces. In *21th International Conference on Very Large Data Bases (VLDB 1995)*, 1995.

[58] Nieves R Brisaboa, Antonio Farina, Oscar Pedreira, and Nora Reyes. Similarity search using sparse pivots for efficient multimedia information retrieval. In *Eighth IEEE International Symposium on Multimedia (ISM'06)*, pages 881–888, Dec 2006.

[59] Walter A. Burkhard and Robert M. Keller. Some approaches to best-match file searching. *Commun. ACM*, 16(4):230–236, April 1973.

[60] Benjamin Bustos, Gonzalo Navarro, and Edgar Chávez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357 – 2366, 2003.

[61] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision - ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792. Springer Berlin Heidelberg, 2010.

[62] Francesco Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36(12):2945 – 2954, 2003.

[63] Vijay Chandrasekhar, Jie Lin, Olivier Morère, Hanlin Goh, and Antoine Veillard. A Practical Guide to CNNs and Fisher Vectors for Image Instance Retrieval. *arXiv preprint arXiv:1508.02496*, 2015.

[64] Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Effective proximity retrieval by ordering permutations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(9):1647–1658, 2008.

[65] Edgar Chávez, Verónica Ludueña, Nora Reyes, and Patricia Roggero. Faster proximity searching with the distal SAT. *Information Systems*, 59:15 – 47, 2016.

[66] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, September 2001.

[67] David Chen, Sam Tsai, Vijay Chandrasekhar, Gabriel Takacs, Huizhong Chen, Ramakrishna Vedantham, Radek Grzeszczuk, and Bernd Girod. Residual enhanced visual vectors for on-device image matching. In *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, pages 850–854, Nov 2011.

[68] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct 2007.

[69] Kenneth L. Clarkson. Nearest-neighbor searching and metric space dimensions. In *In Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*. MIT Press, 2006.

[70] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[71] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[72] Richard Connor. Mir-flickr near-duplicate data. `mir-flickr-near-duplicates.appspot.com`, January 2015.

[73] Richard Connor, Franco Alberto Cardillo, Robert Moss, and Fausto Rabitti. Evaluation of jensen-shannon distance over sparse data. In Nieves Brisaboa, Oscar Pedreira, and Pavel Zezula, editors, *Similarity Search and Applications: 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings*, pages 163–168. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[74] Richard Connor, Franco Alberto Cardillo, Lucia Vadicamo, and Fausto Rabitti. Hilbert Exclusion: Improved metric search through finite isometric embeddings. *ACM Transactions on Information Systems*, 35(3):17:1–17:27, December 2016.

[75] Richard Connor and Robert Moss. A multivariate correlation distance for vector spaces. In Gonzalo Navarro and Vladimir Pestov, editors, *Similarity Search and Applications*, volume 7404 of *Lecture Notes in Computer Science*, pages 209–225. Springer Berlin Heidelberg, 2012.

[76] Richard Connor, Lucia Vadicamo, Franco Alberto Cardillo, and Fausto Rabitti. *Supermetric Search with the Four-Point Property*, pages 51–64. Springer International Publishing, 2016.

[77] Richard Connor, Lucia Vadicamo, Franco Alberto Cardillo, and Fausto Rabitti. *Supermetric search*. 2018.

[78] Richard Connor, Lucia Vadicamo, and Fausto Rabitti. *High-Dimensional Simplexes for Supermetric Search*, pages 96–109. Springer International Publishing, 2017.

[79] RRichard Connor, Franco Alberto Cardillo, Stewart MacKenzie-Leigh, and Robert Moss. Identification of mir-flickr near-duplicate images. In *10th International Conference on Computer Vision Theory and Applications*. 2015.

[80] Douglas E. Critchlow. *Metric Methods for Analyzing Partially Ranked Data*, volume 34 of *Lecture Notes in Statistics*. 1985.

[81] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. *Workshop on statistical learning in computer vision, ECCV*, 1(1-22):1–2, 2004.

[82] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2012.

[83] Sanjoy Dasgupta and Philip M. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555 – 569, 2005. Special Issue on COLT 2002.

[84] Ritendra Datta, Jia Li, and James Z. Wang. Content-based image retrieval: Approaches and trends of the new age. In *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*, MIR '05, pages 253–262, 2005.

[85] Vin De Silva and Joshua B Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, 2004.

[86] J. M. De Tilly. Essai de geometrie analytique generale. *Memoires couronnes et autresmemo ires publies par l'Academie Royale de Belgique, 47 (1892-3),memoire 5.*, 1892.

[87] Jonathan Delhumeau, Philippe-Henri Gosselin, Hervé Jégou, and Patrick Pérez. Revisiting the VLAD image representation. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM 2013, pages 653–656, New York, NY, USA, 2013. ACM.

[88] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, June 2009.

[89] Yining Deng and B.S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, Aug 2001.

[90] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2016.

[91] Persi Diaconis. Group representations in probability and statistics. *Lecture Notes-Monograph Series*, pages i–192, 1988.

[92] Vlastislav Dohnal. Indexing structures for searching in metric spaces. PhD thesis. Faculty of Informatics, Masaryk University in Brno, Czech Republic, 2004.

[93] Vlastislav Dohnal, Claudio Gennaro, Pasquale Savino, and Pavel Zezula. D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications*, 21(1):9–33, 2003.

[94] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[95] Dominik Maria Endres and Johannes E. Schindelin. A new metric for probability distributions. *Information Theory, IEEE Transactions on*, 49(7):1858–1860, 2003.

[96] Andrea Esuli. Use of permutation prefixes for efficient and scalable approximate similarity search. *Information Processing & Management*, 48(5):889 – 902, 2012.

[97] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 28–36, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[98] Fabrizio Falchi. Visual information retrieval library.

[99] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. Approximate nearest neighbor searching in multimedia databases. In *Proceedings 17th International Conference on Data Engineering*, pages 503–511, 2001.

[100] Karina Figueroa and Kimmo Fredriksson. Speeding up permutation based indexing with indexing. In *2009 Second International Workshop on Similarity Search and Applications*, pages 107–114, Aug 2009.

[101] Karina Figueroa, Gonzalo Navarro, and Edgar Chávez. Metric spaces library. `www.sisap.org/library/manual.pdf`, 2007.

[102] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[103] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[104] Bent Fuglede and Flemming Topsøe. Jensen-shannon divergence and hilbert space embedding. In *IEEE International Symposium on Information Theory*, pages 31–31, 2004.

[105] Keinosuke Fukunaga. Intrinsic dimensionality extraction. In *Classification Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 347 – 360. Elsevier, 1982.

[106] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[107] Paul Gaiha and S. K. Gupta. Adjacent vertices on a permutohedron. *SIAM Journal on Applied Mathematics*, 32(2):323–327, 1977.

[108] Dorian Galvez-Lopez and Juan D. Tardos. Real-time loop detection with bags of binary words. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 51–58, Sept 2011.

[109] Claudio Gennaro, Giuseppe Amato, Paolo Bolettieri, and Pasquale Savino. An approach to content-based image retrieval based on the lucene search engine library. In Mounia Lalmas, Joemon Jose, Andreas Rauber, Fabrizio Sebastiani, and Ingo Frommholz, editors, *Research and Advanced Technology for Digital Libraries*, volume 6273 of *Lecture Notes in Computer Science*, pages 55–66. Springer Berlin Heidelberg, 2010.

[110] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587, June 2014.

[111] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293 – 306, 1985.

[112] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[113] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254, 2017.

[114] Costantino Grana, Daniele Borghesani, Marco Manfredi, and Rita Cucchiara. A fast approach for integrating ORB descriptors in the bag of words model. In Cees G. M. Snoek, Lyndon S. Kennedy, Reiner Creutzburg, David Akopian, Dietmar Wüller, Kevin J. Matherson, Todor G. Georgiev, and Andrew Lumsdaine, editors, *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, March 2013.

[115] Robert M. Gray and David L. Neuhoff. Quantization. *Information Theory, IEEE Transactions on*, 44(6):2325–2383, Oct 1998.

[116] Sven Grewenig, Joachim Weickert, and Andrés Bruhn. *From Box Filtering to Fast Explicit Diffusion*, pages 533–542. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[117] F. Guillet and H.J. Hamilton. *Quality Measures in Data Mining*. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2007.

[118] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27 – 48, 2016. Recent Developments on Deep Big Vision.

[119] Gaëtan Hadjeres and Frank Nielsen. Deep rank-based transposition-invariant distances on musical sequences. *arXiv preprint arXiv:1709.00740*, 2017.

[120] James Hafner, Harpreet S. Sawhney, William Equitz, Myron Flickner, and Wayne Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE transactions on pattern analysis and machine intelligence*, 17(7):729–736, 1995.

[121] Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[122] Jonathon S Hare, Paul H Lewis, Peter GB Enser, and Christine J Sandom. Mind the gap: Another look at the problem of the semantic gap in image retrieval. 2006.

[123] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[124] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[125] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[126] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Comparative evaluation of binary features. In *Computer Vision - ECCV 2012*, Lecture Notes in Computer Science, pages 759–773. Springer Berlin Heidelberg, 2012.

[127] Magnus Lie Hetland. *The Basic Principles of Metric Indexing*, pages 199–232. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[128] Magnus Lie Hetland. Ptolemaic indexing. *Journal of Computational Geometry*, 6(1):165–184, 2015.

[129] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[130] Geoffrey E Hinton. To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–547, 2007.

[131] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[132] Gísli R Hjaltason and Hanan Samet. Incremental similarity search in multimedia databases. Technical report, 2000.

[133] Gisli R. Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces (survey article). *ACM Trans. Database Syst.*, 28(4):517–580, December 2003.

[134] Alston S. Householder. *The Theory of Matrices in Numerical Analysis*. A Blaisdell book in pure and applied sciences: introduction to higher mathematics. Blaisdell Publishing Company, 1964.

[135] Mark J. Huiskes and Michael S. Lew. The MIR Flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008. ACM.

[136] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[137] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1998.

[138] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision–ECCV 2012*, volume 7573 of *Lecture Notes in Computer Science*, pages 774–787. Springer, 2012.

[139] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317. Springer, oct 2008.

[140] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Packing bag-of-features. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2357 –2364, 29 2009-oct. 2 2009.

[141] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87:316–336, May 2010.

[142] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128, Jan 2011.

[143] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, jun 2010.

[144] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sànchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, Sept 2012.

[145] Hervé Jégou, Cordelia Schmid, Hedi Harzallah, and Jakob Verbeek. Accurate image search using the contextual dissimilarity measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):2–11, Jan 2010.

[146] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[147] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[148] Ian Jolliffe. *Principal Component Analysis*. John Wiley & Sons, Ltd, 2014.

[149] Iraj Kalantari and Gerard McDonald. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, SE-9(5):631–634, Sept 1983.

[150] Junzo Kamahara, Takashi Nagamatsu, and Naoki Tanaka. Conjunctive ranking function using geographic distance and image distance for geotagged image retrieval. In *Proceedings of the ACM Multimedia 2012 Workshop on Geotagging and Its Applications in Multimedia*, GeoMM '12, pages 9–14, New York, NY, USA, 2012. ACM.

[151] Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. In *An introduction to L1-norm based statistical data analysis*, volume 5 of *Computational Statistics & Data Analysis*, 1987.

[152] Maurice George Kendall. Rank correlation methods. 1948.

[153] Saluka Ranasinghe Kodituwakku and S. Selvarajah. Comparison of color features for image retrieval. *Indian Journal of Computer Science and Engineering*, 1(3):207–211, 2004.

[154] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Modeling spatial layout with Fisher Vectors for image categorization. In *ICCV 2011 - International Conference on Computer Vision*, pages 1487–1494, Barcelona, Spain, November 2011.

[155] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[156] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[157] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[158] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, 2006.

[159] Lucien Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer My Copy UK, 1986.

[160] Yann Le Cun, LD Jackel, B Boser, JS Denker, HP Graf, I Guyon, D Henderson, RE Howard, and W Hubbard. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

[161] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[162] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[163] Suwon Lee, SuGil Choi, and Hyun S Yang. Bag-of-binary-features for fast image representation. *Electronics Letters*, 51(7):555–557, 2015.

[164] Michael KK Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.

[165] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, Nov 2011.

[166] Gil Levi and Tal Hassner. LATCH: Learned Arrangements of Three Patch Codes. *CoRR*, abs/1501.03719, 2015.

[167] Xirong Li, Tiberio Uricchio, Lamberto Ballan, Marco Bertini, Cees G. M. Snoek, and Alberto Del Bimbo. Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval. *ACM Comput. Surv.*, 49(1):14:1–14:39, June 2016.

[168] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep learning of binary hash codes for fast image retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[169] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[170] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262 – 282, 2007.

[171] Stuart Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982.

[172] Jakub Lokoč, Magnus Lie Hetland, Tomáš Skopal, and Christian Beecks. Ptolemaic indexing of the signature quadratic form distance. In *Proceedings of the Fourth International Conference on SImilarity Search and APplications*, pages 9–16. ACM, 2011.

[173] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[174] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd International Conference Very Large Data Bases*, VLDB '07, pages 950–961, Vienna, Austria, 2007.

[175] Bangalore S Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7: multimedia content description interface*, volume 1. John Wiley & Sons, 2002.

[176] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[177] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004. British Machine Vision Computing 2002.

[178] Jiří Matoušek. *Lectures on Discrete Geometry*. Graduate Texts in Mathematics. Springer New York, 2013.

[179] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[180] Geoffrey McLachlan and David Peel. *Finite Mixture Models*. Wiley series in probability and statistics. Wiley, 2000.

[181] Rajiv Mehrotra and James E. Gary. Similar-shape retrieval in shape data management. *Computer*, 28(9):57–62, Sep 1995.

[182] Karl Menger. Untersuchungen über allgemeine metrik. *Mathematische Annalen*, 100:75–163, 1928.

[183] Vladimir Mic, David Novak, and Pavel Zezula. Improving sketches for similarity search. In *Proceedings of MEMICS 2015*, pages 45–57, 2015.

[184] María Luisa Micó, José Oncina, and Enrique Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.*, 15(1):9–17, January 1994.

[185] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615 –1630, oct. 2005.

[186] Ondrej Miksik and Krystian Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2681–2684, Nov 2012.

[187] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of CNN advances on the imagenet. *CoRR*, abs/1606.02228, 2016.

[188] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

[189] Hisham Mohamed and Stéphane Marchand-Maillet. Metric suffix array for large-scale similarity search. *INVITED SPEAKER: Analyzing the Performance of Top-K Retrieval Algorithms*, page 1989, 2013.

[190] Bilegsaikhan Naidan, Leonid Boytsov, and Eric Nyberg. Permutation search methods are efficient, yet faster search is possible. *Proc. VLDB Endow.*, 8(12):1618–1629, August 2015.

[191] Gonzalo Navarro. Searching in metric spaces by spatial approximation. *The VLDB Journal*, 11(1):28–46, 2002.

[192] Gonzalo Navarro, Rodrigo Paredes, Nora Reyes, and Cristian Bustos. An empirical evaluation of intrinsic dimension estimators. *Information Systems*, 64(Supplement C):206 – 218, 2017.

[193] Gonzalo Navarro and Nora Reyes. *String Processing and Information Retrieval: 9th International Symposium, SPIRE 2002 Lisbon, Portugal, September 11–13, 2002 Proceedings*, chapter Fully Dynamic Spatial Approximation Trees, pages 254–270. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[194] Gonzalo Navarro and Nora Reyes. Dynamic spatial approximation trees. *J. Exp. Algorithmics*, 12:1.5:1–1.5:68, June 2008.

[195] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168, 2006.

[196] Xia-mu Niu and Yu-hua Jiao. An overview of perceptual hashing. *Acta Electronica Sinica*, 36(7):1405–1411, 2008.

[197] Hartmut Noltemeier, Knut Verbarg, and Christian Zirkelbach. *Monotonous Bisector* Trees — a tool for efficient partitioning of complex scenes of geometric objects*, pages 186–203. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[198] David Novak, Michal Batko, and Pavel Zezula. Metric index: An efficient and scalable solution for precise and approximate similarity search. *Information Systems*, 36(4):721–733, 2011.

[199] David Novak, Michal Batko, and Pavel Zezula. Large-scale image retrieval using neural net descriptors. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1039–1040. ACM, 2015.

[200] David Novak and Pavel Zezula. *PPP-Codes for Large-Scale Similarity Searching*, pages 61–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[201] Ferdinand Österreicher and Igor Vajda. A new class of metric divergences on probability spaces and and its statistical applications. *Ann. Inst. Statist. Math.*, 55:639–653, 2003.

[202] Mira Park, Jesse S Jin, and Laurence S Wilson. Fast content-based image retrieval using quasi-gabor filter and reduction of image feature dimension. In *Proceedings Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 178–182, 2002.

[203] Marco Patella and Paolo Ciaccia. Approximate similarity search: A multi-faceted problem. *Journal of Discrete Algorithms*, 7(1):36 – 48, 2009. Selected papers from the 1st International Workshop on Similarity Search and Applications (SISAP)1st International Workshop on Similarity Search and Applications (SISAP).

[204] Xiaojiang Peng, Limin Wang, Yu Qiao, and Qiang Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014*, volume 8691 of *Lecture Notes in Computer Science*, pages 660–674. Springer International Publishing, 2014.

[205] Michal Perd'och, Ondrej Chum, and Jiri Matas. Efficient representation of local geometry for large scale object retrieval. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 9–16, June 2009.

[206] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[207] Florent Perronnin and Diane Larlus. Fisher Vectors Meet Neural Networks: A Hybrid Classification Architecture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3743–3752, 2015.

[208] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384–3391, June 2010.

[209] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 143–156. Springer Berlin Heidelberg, 2010.

[210] Vladimir Pestov. Indexability, concentration, and vc theory. *Journal of Discrete Algorithms*, 13:2–18, 2012.

[211] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR 2007. IEEE Conference on*, pages 1–8, 2007.

[212] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[213] Alessandro Pieropan, Mårten Björkman, Niklas Bergström, and Danica Kragic. Feature descriptors for tracking by detection: a benchmark. *CoRR*, abs/1607.06178, 2016.

[214] Ives Renê Venturini Pola, Caetano Traina, and Agma Juci Machado Traina. The nobh-tree: Improving in-memory metric access methods by using metric hyperplanes with non-overlapping nodes. *Data & Knowledge Engineering*, 94, Part A:65 – 88, 2014.

[215] Gill A. Pratt. Is a cambrian explosion coming for robotics? *Journal of Economic Perspectives*, 29(3):51–60, August 2015.

[216] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 1137–1144, Cambridge, MA, USA, 2006. MIT Press.

[217] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[218] Ali Sharif Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *CoRR*, abs/1412.6574, 2014.

[219] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[220] Edward Rosten and Tom Drummond. *Machine Learning for High-Speed Corner Detection*, pages 430–443. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[221] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, Nov 2011.

[222] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[223] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[224] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[225] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[226] Jorge Sànchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.

[227] Jorge Sànchez and Javier Redolfi. Exponential family fisher vector for image classification. *Pattern Recognition Letters*, 59:26 – 32, 2015.

[228] Joe Santmyer. For all possible distances look to the permutohedron. *Mathematics Magazine*, 80(2):pp. 120–125, 2007.

[229] Isaac J. Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, 39(4):811–841, 1938.

[230] Sebastian Scholtes. A characterisation of inner product spaces by the maximal circumradius of spheres. *Archiv der Mathematik*, 101(3):235–241, 2013.

[231] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep fisher networks for large-scale image classification. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 163–171. Curran Associates, Inc., 2013.

[232] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[233] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1470–1477, 2003.

[234] Matthew Skala. Counting distance permutations. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 362–369, April 2008.

[235] Tomáš Skopal and Benjamin Bustos. On nonmetric similarity search problems in complex domains. *ACM Comput. Surv.*, 43(4):34:1–34:50, October 2011.

[236] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, December 2000.

[237] Stephen M. Smith and J. Michael Brady. Susan—a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[238] Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, Ioannis Yiannis Kompatsiaris, Grigorios Tsoumakas, and Ioannis Vlahavas. A comprehensive study over VLAD and product quantization in large-scale image retrieval. *Multimedia, IEEE Transactions on*, 16(6):1713–1728, 2014.

[239] Vladyslav Sydorov, Mayu Sakurada, and Christoph H. Lampert. Deep fisher kernels - end to end learning of the fisher kernel gmm parameters. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[240] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[241] Eric Sadit Tellez, Edgar Chávez, and Gonzalo Navarro. Succinct nearest neighbor search. *Information Systems*, 38(7):1019 – 1030, 2013.

[242] Bart Thomee, Erwin M Bakker, and Michael S Lew. TOP-SURF: A visual words toolkit. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 1473–1476. ACM, 2010.

[243] Bart Thomee, Benjamin Elizalde, David A Shamma, Karl Ni, Gerald Friedland, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[244] Giorgos Tolias and Yannis Avrithis. Speeded-up, relaxed spatial matching. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1653–1660, Nov 2011.

[245] Giorgos Tolias, Teddy Furon, and Hervé Jégou. Orientation covariant aggregation of local descriptors with embeddings. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pages 382–397. Springer International Publishing, 2014.

[246] Giorgos Tolias and Hervé Jégou. Local visual query expansion: Exploiting an image collection to refine local descriptors. Research Report RR-8325, July 2013.

[247] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. *CoRR*, abs/1511.05879, 2015.

[248] Flemming Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory*, 46(4):1602–1609, Jul 2000.

[249] Flemming Topsøe. Jensen-Shannon divergence and norm-based measures of discrimination and variation. Technical report, 2003.

[250] Tinne Tuytelaars, Krystian Mikolajczyk, et al. Local invariant feature detectors: A survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2008.

[251] Yusuke Uchida and Shigeyuki Sakazawa. Image retrieval with fisher vectors of binary features. In *Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on*, pages 23–28, Nov 2013.

[252] Jeffrey K Uhlmann. Satisfying general proximity / similarity queries with metric trees. *Information Processing Letters*, 40(4):175 – 179, 1991.

[253] Shimon Ullman. *High-Level Vision - Object Recognition and Visual Cognition*. MIT Press, July 1996.

[254] Tiberio Uricchio, Marco Bertini, Lorenzo Seidenari, and Alberto Del Bimbo. Fisher encoded convolutional bag-of-windows for efficient image retrieval and social image tagging. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.

[255] Jan C. van Gemert, Jan-Mark Geusebroek, Cor J. Veenman, and Arnold W.M. Smeulders. Kernel codebooks for scene categorization. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision - ECCV 2008*, volume 5304 of *Lecture Notes in Computer Science*, pages 696–709. Springer Berlin Heidelberg, 2008.

[256] Jan C. Van Gemert, Cor J. Veenman, Arnold W.M. Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1271–1283, July 2010.

[257] Dominik Van Opdenbosch, Georg Schroth, Robert Huitl, Sebastian Hilsenbeck, Adrian Garcea, and Eckehard Steinbach. Camera-based indoor positioning using scalable streaming of compressed binary image signatures. In *IEEE International Conference on Image Processing*, 2014.

[258] Enrique Vidal Ruiz. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, 4(3):145 – 157, 1986.

[259] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 157–166, New York, NY, USA, 2014. ACM.

[260] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367, June 2010.

[261] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[262] Wallace A. Wilson. A relation between metric and euclidean spaces. *American Journal of Mathematics*, 54(3):505–517, 1932.

[263] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[264] Chee Sun Won, Dong Kwon Park, and Soo-Jun Park. Efficient use of mpeg-7 edge histogram descriptor. *ETRI journal*, 24(1):23–30, 2002.

[265] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801, June 2009.

[266] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 311–321. Society for Industrial and Applied Mathematics, 1993.

[267] Peter N Yianilos. Excluded middle vantage point forests for nearest neighbor search. In *In DIMACS Implementation Challenge, ALENEX'99*, 1999.

[268] Joe Yue-Hei Ng, Fan Yang, and Larry S Davis. Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 53–61, 2015.

[269] Joe Yue-Hei Ng, Fan Yang, and Larry S. Davis. Exploiting local features from deep networks for image retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[270] Matthew D. Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*, pages 818–833. Springer International Publishing, Cham, 2014.

[271] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

[272] Zheng-Jun Zha, Linjun Yang, Tao Mei, Meng Wang, Zengfu Wang, Tat-Seng Chua, and Xian-Sheng Hua. Visual query suggestion: Towards capturing user intent in internet image search. *ACM Trans. Multimedia Comput. Commun. Appl.*, 6(3):13:1–13:19, August 2010.

[273] Xiao Zhang, Zhiwei Li, Lei Zhang, Wei-Ying Ma, and Heung-Yeung Shum. Efficient indexing for large scale visual search. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1103–1110, Sept 2009.

[274] Yu Zhang, Chao Zhu, Stephane Bres, and Liming Chen. Encoding local binary descriptors by bag-of-features with hamming distance for visual object categorization. In Pavel Serdyukov, Pavel Braslavski, SergeiO. Kuznetsov, Jaap Kamps, Stefan Rüger, Eugene Agichtein, Ilya Segalovich, and Emine Yilmaz, editors, *Advances in Information Retrieval*, volume 7814 of *Lecture Notes in Computer Science*, pages 630–641. Springer Berlin Heidelberg, 2013.

[275] Wan-Lei Zhao, Hervé Jégou, and Guillaume Gravier. Oriented pooling for dense and non-dense rotation-invariant features. In *BMVC - 24th British Machine Vision Conference*, September 2013.

[276] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014.

[277] Wengang Zhou, Houqiang Li, and Qi Tian. Recent advance in content-based image retrieval: A literature survey. *CoRR*, abs/1706.06064, 2017.

[278] Xiang Sean Zhou and Thomas S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8(6):536–544, Apr 2003.

[279] Y. T. Zhou and Rama Chellappa. Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks*, volume 1998, pages 71–78, 1988.

[280] Günter M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer New York, 1995.