LEARNING THE CONSENSUS OF MULTIPLE CORRESPONDENCES BETWEEN DATA STRUCTURES

Carlos Francisco Moreno García

Carlos Francisco Moreno García

LEARNING THE CONSENSUS OF MULTIPLE CORRESPONDENCES BETWEEN DATA STRUCTURES

DOCTORAL THESIS

Supervised by:

Dr. Francesc Serratosa Casanelles

Department of Computer Science and Mathematics

UNIVERSITAT ROVIRA I VIRGILI

Tarragona, Spain

2016

This work has been supported by a Ph. D. grant conceded by Consejo Nacional de Ciencia y Tecnología (CONACyT Mexico), and by Comisión Interministerial de Ciencia y Tecnología (CICYT Spain) projects TIN2013-47245-C2-2-R and DPI2013-42458-P.

UNIVERSITAT ROVIRA I VIRGILI

School of Engineering (ETSE)
Department of Computer Science and Mathematics
Av. Països Catalans, 26
43007 Tarragona
Tarragona, Spain
Tel. 977 337 878

I STATE that the present study, entitled: "Learning the Consensus of Multiple Correspondences between Data Structures", presented by **Carlos Francisco Moreno García**, for the award of the degree of Doctor, has been carried out under my supervision at the Department of Computer Science and Mathematics of this University, and that it fulfils all the requirements to be eligible for the PhD International Mention.

Tarragona, Spain
May 10th, 2016

[illegible]

Dr. Francesc Serratosa Casanelles

Agradecimientos

Se dice que somos "como enanos a hombros de gigantes". En el caso de este trabajo doctoral, considero que hay mucha razón en emplear estas palabras, ya que el producto de la investigación que he realizado a lo largo de estos tres años está fuertemente ligado al apoyo que he recibido por parte de las siguientes personas, por los cuales siento un enorme respeto y admiración. A continuación, me gustaría dedicar unas palabras a los siguientes gigantes.

En primer lugar, quisiera agradecer al Dr. Francesc Serratosa por haber confiado en mí y permitirme ser parte de su grupo de investigación. Su constante ayuda y consejo han sido de vital importancia para el desarrollo de todos mis proyectos a lo largo de mi maestría y doctorado. Asimismo, me gustaría expresar mi gratitud hacia mi compañero de grupo, Xavier Cortés, quien siempre me proporcionó su incondicional ayuda, y a quien deseo lo mejor en su defensa doctoral. Ha sido para mí un auténtico placer poder pertenecer a este grupo de trabajo que, aunque pequeño, siempre estuvo al nivel de las exigencias.

Gracias al Prof. Xiaoyi Jiang y a todos los miembros de su grupo de investigación "Computer Vision and Pattern Recognition" de la Universidad de Münster, por su amable recibimiento durante los tres meses de mi estancia doctoral. Este periodo fue muy provechoso para mí, tanto a nivel profesional como personal.

Una mención especial a todos los amigos que a lo largo de estos años he conocido en Tarragona. Ustedes también son parte de mis éxitos, y aunque la naturaleza de este mundo nos evoca a ir de ciudad en ciudad y convivir por poco tiempo, me complace saber que tengo amistades en muchas partes del planeta.

También quiero dar las gracias a mi familia, que con tanto amor y comprensión ha sido la base de este trabajo, y que, a pesar de la distancia, ha estado siempre junto a mí. A los pilares de mi vida, mis padres (Laura y Carlos), así como mis abuelos (Luz y Carlos) y mis tíos (Adriana, Eduardo y Jorge). Todos ellos han sido en buena parte mis profesores de vida, y son los valores que me han inculcado los que me han llevado hasta este lugar.

Finalmente, me es imprescindible dedicar esta tesis a la persona que me ha acompañado durante esta aventura, y que espero lo siga haciendo por el resto de mi vida. Gracias Maga, no existen palabras suficientes para decirte lo mucho que agradezco tu incesante papel como coautora, confidente, amiga, y muchísimo más. Admiro la fuerza que has tenido para ser mi apoyo y a la vez haber trabajado muy duro para alcanzar nuestra meta en común, y quién sabe, algún día no tan lejano, me gustaría que estas palabras y las tuyas pudieran ser la inspiración de alguien más.

Resumen

Una correspondencia se define como el resultado de una función biyectiva que estipula un conjunto de asignaciones entre los elementos de dos estructuras de datos, tales como conjuntos de puntos, cadenas, árboles, grafos y clústeres de datos. Éste es un proceso referido comúnmente como "matching". El principal objetivo de generar correspondencias entre un par de estructuras de datos es encontrar la similitud entre éstas, lo cual conlleva a estudios más avanzados relacionados con el reconocimiento, la clasificación y el análisis estadístico. Aunque el uso de las correspondencias ofrece ventajas tales como una estructura simple y la enorme disponibilidad de algoritmos para su generación, se han hecho pocos esfuerzos por usarlas como estructuras de datos *per se* y así, llevar a cabo tareas de reconocimiento de patrones y visión por computadora directamente con éstas.

En esta tesis, presentamos un marco de trabajo para aprender el consenso dadas múltiples correspondencias. Se asume que las distintas partes involucradas han generado dichas correspondencias por separado, y nuestro sistema actúa como un mecanismo que calibra distintas características y considera diferentes parámetros para aprender las mejores asignaciones y así, conformar una correspondencia con la mayor precisión posible a expensas de un costo computacional razonable. El marco de trabajo de consenso es presentado en una forma gradual, comenzando por los acercamientos más básicos que utilizaban exclusivamente conceptos bien definidos o únicamente un par de correspondencias, hasta el modelo final que es capaz de considerar múltiples correspondencias, con la capacidad de aprender automáticamente algunos parámetros de ponderación. Cada paso de este marco de trabajo es evaluado usando bases de datos de naturaleza variada para demostrar efectivamente que es posible tratar diferentes escenarios de matching.

Adicionalmente, dos avances suplementarios relacionados con correspondencias son presentados en este trabajo. En primer lugar, una nueva métrica de distancia para correspondencias ha sido desarrollada, la cual derivó en una nueva estrategia para la búsqueda de medias ponderadas. En segundo lugar, un marco de trabajo específicamente diseñado para generar correspondencias en el campo del registro de imágenes ha sido establecida, donde se considera que una de las imágenes es una imagen completa, y la otra es una muestra pequeña de ésta. La conclusión presenta nuevas percepciones de cómo nuestro marco de trabajo de consenso puede ser mejorada, y cómo los dos desarrollos paralelos pueden converger con éste.

Abstract

We define a correspondence as the result of a bijective function that assigns a set of mappings between the elements of two data structures such as a set of points, strings, trees, graphs or data clusters. This is a process often referred as matching. The main purpose of generating correspondences between a pair of data structures is to find a similarity between them, which can lead to further studies related to recognition, classification or statistical analysis. Despite correspondences offering several advantages, such as their simple structure and the enormous availability of algorithms for their generation, little effort has been devoted to use them as data structures *per se* and so, perform pattern recognition and computer vision tasks directly with them.

In this work, we present a framework to learn the consensus given multiple correspondences. It is assumed that the several parties involved have generated separately these correspondences, and our system acts as a mechanism that gauges several characteristics and considers different parameters to learn the best mappings and thus, conform a correspondence with the highest possible accuracy at the expense of a reasonable computational cost. The consensus framework is presented in a gradual form, starting from the most basic approaches that used exclusively well-known concepts or only two correspondences, until the final model which is able to consider multiple correspondences, with the capability of automatically learning some weighting parameters. Each step of the framework is evaluated using databases of varied nature to effectively demonstrate that it is capable to address different matching scenarios.

In addition, two supplementary advances related to correspondences are presented in this work. Firstly, a new distance metric for correspondences has been developed, which lead to a new strategy for the weighted mean correspondence search. Secondly, a framework specifically designed for correspondence generation in the image registration field has been established, where it is considered that one of the images is a full image, and the other one is a small sample of it. The conclusion presents insights of how our consensus framework can be enhanced, and how these two parallel developments can converge with it.

Contents

Chapter 1 Introduction 1
1.1 What are Correspondences? 3
1.2 Consensus Correspondence 5
1.3 Thesis Structure 7
Chapter 2 Basic Definition 9
2.1 The Concept of Distance 11
2.2 Correspondences and Matching Algorithms 11
2.3 Weighted Mean and Median 25
2.4 Consensus Frameworks 28
Chapter 3 Generalised Median Computation of Multiple Correspondences based on the Hamming Distance 31
3.1 Introduction 33
3.2 Definition and Notations 33
3.3 Methodology 34
3.4 Experimental Validation 39
3.5 Discussion 42
Chapter 4 Weighted Mean Consensus of a Pair of Correspondences 45
4.1 Introduction 47
4.2 Definitions and Notations 47
4.3 Methodology 49
4.4 Reasoning about Optimality 51
4.5 Experimental Validation 54
4.6 Discussion 62
Chapter 5 Consensus of a Pair of Correspondences through the Bipartite Framework 65
5.1 Introduction 67
5.2 Definitions and Notations 67

5.3 Methodology........68
5.4 Reasoning about Optimality........71
5.5 Experimental Validation........71
5.6 Discussion........77
Chapter 6 Consensus of Multiple Correspondences........79
6.1 Introduction........81
6.2 Definitions and Notations........82
6.3 Methodology........84
6.4 Experimental Validation........89
6.5 Discussion........101
Chapter 7 Learning Weights for the Consensus of Multiple Correspondences........103
7.1 Introduction........105
7.2 Definitions and Notations........105
7.3 Methodology........106
7.4 Experimental Validation........108
7.5 Discussion........118
Chapter 8 A New Distance and Weighted Mean Search Strategy for Correspondences........119
8.1 Introduction........121
8.2 Methodology........123
8.3 Experimental Validation........129
8.4 Discussion........137
Chapter 9 Partial to Full Correspondence Generation for Image Registration........139
9.1 Introduction........141
9.2 Definitions and Notations........142
9.3 Methodology........144
9.4 Reasoning about Complexity and Speedup........146
9.5 Experimental Validation........147

9.6 Discussion 155
Chapter 10 Conclusions and Future Work 157
10.1 Conclusions 159
10.2 Future Work 160
List of Publications 161
References 165

Chapter 1 Introduction

1.1 What are Correspondences?

We define a *correspondence* as the result of a bijective function that assigns a set of *mappings* between the elements of two *data structures*. This assignment is made in a directional way from an output data structure to an input data structure. Moreover, data structures are defined as particular and well-defined ways of organising data in computer science, such as a set of points, strings, trees, graphs or data clusters (a data cluster refers in this case to a set of elements grouped into two or more classes). In recent years, several works have been devoted to convert objects generated by humans (i.e. sketches, handwriting, markers in an image) into data structures, extending their use in pattern recognition and computer vision tasks.

Each local element of a data structure (a point, if the object is represented as a set of points; a character, if it is represented as a string; or a node and its neighbouring edges, if it is represented as a tree or graph) has some attributes that give local information of the data structure. The main purpose of generating correspondences is to find a similarity between them, which can lead to further studies related to recognition, classification or statistical analysis. An example of the representation of a correspondence is provided in figure 1.1.

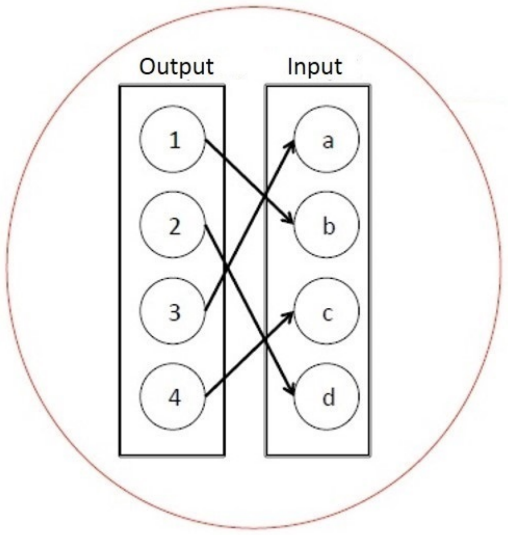


Figure 1. 1. A correspondence with 4 mappings between an output and an input data structure.

Using the correspondences instead of the structures that are being mapped provides several advantages. Firstly, correspondences are the most intuitive form of pattern recognition from a human perspective. Most of people are not able to directly correlate a data structure, such as a tree or a graph, with the object that these are supposed to represent. Nonetheless, it is more viable that a person observes the mappings between the structures and acquires an intuition of the relation between them. Secondly, correspondences can be generated manually by connecting two sets of elements in an instinctive form. This is an advantage that could not be attained by, for instance, using graphs, since a graph generated manually would be clearly distinct from a graph generated through the use of a computational algorithm and thus, both structures would not be comparable. In contrast, it is possible to compare a manual correspondence with an automatic one, given that only an analysis of the mappings is needed. Thirdly, correspondences offer a simple computational structure, which consists of an array of numbers that indicates which elements from the output object are related to which element of the input object. These particularity makes correspondences very suitable to be used in software which relies on simple mathematical computations, such as Octave or Matlab. Finally, correspondences can be used on most of the data structures studied in pattern recognition. This translates into a very varied application domain for the use of correspondence.

In relation to this last point, correspondences have a significant number of uses; the most common one being image recognition. It is possible to represent through correspondences very diverse types of image repositories [1], [2], [3], [4], [5], as long as sufficient features can be extracted from the contained images. Most notably in the case of biometrics [6], a correspondence can be generated between virtually any part of the human body, such as a pair of skin portions [7] of fingerprints [8], palmprints [9], [10], [11], [12], faces [13], etc. It is even possible to generate a correspondence through the whole structure of the body for pose recognition and tracking [14], [15]. Furthermore, correspondences can aid on additional types of recognition. Just to name a few, work has been done in hand-written electronic scheme digitalisation [16], patterns of chemical data [17], [18], proteins, letters and architectural designs [19]. In addition, it is of special interest to highlight the use of correspondences on social networks given the thriving development of this technology [20], [21]. These proposals are oriented to recognise a person based on their social media activity. In these cases, if two graphs represent the cloud of users in two different social networks, simply by analysing a node's behaviour a correspondence can be generated between nodes of different graphs and thus, a subject can be identified without specifically knowing the personal information of the subject represented by such nodes.

As mentioned before, correspondences can be generated either by manual or automatic methods. In the case that correspondences are obtained through an automatic algorithm, the process is most commonly referred in the literature as *matching*. Several automatic matching methods have been presented related to set of points matching [22], [23], [24], string matching [25], tree matching and graph matching [26], [27] [28]. While literature demonstrates that there has been a long standing effort to increase the quality of the methods that perform matching, the aim of the present work is to assume that we encounter several parties which already have applied different matching techniques (manual or automatic), and thus, have already produced multiple correspondences. Therefore, our main interest is not to conform the correspondences, but to collect them and learn a consensus from them.

1.2 Consensus Correspondence

The most practical form to understand the difficulties of learning a consensus correspondence is through an image recognition scenario. For the case that is presented in figure 1.2, there are two pairs of images from the same object, with a different orientation and zoom. Two parties "a" and "b" decide to generate a correspondence that relates these two images. As mentioned before, the process of generating a correspondence can be performed either manually or automatically, which can lead to one or more of the following reasons for discrepancy: 1) One of the parties gives more importance to some of the element's attributes, and the counterpart believes other ones are more important. If the scenario consists only of correspondences generated manually, the strategies are based on the experience of the human specialist. For instance, if elements represent regions of segmented images, one subject may think the area is more important than the colour, and the other one may think the opposite. Conversely, if the scenario is based solely on an automatic method, these differences are gauged by the features or the weights on these features. In a mixed scenario, correspondences would have even less similarities. 2)

Another factor that may influence the correspondence generation in automatic matching methods could be that if the elements' assignment problem is computed with a suboptimal algorithm, different non-exact mappings may appear.

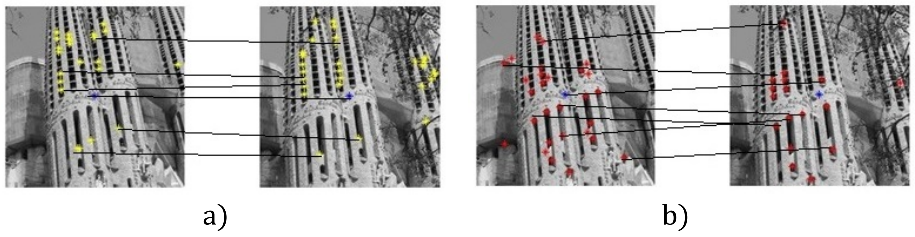


Figure 1. 2. Two images and two correspondences between them.

If figures 1.1(a) and 1.1(b) are compared, given that the sets of salient points extracted from the images are different, the obtained correspondences are even more distinct than if the two parties had already received the points and were only given the task of generating the correspondences.

Regardless of the situation, the aim of a properly learnt consensus correspondence is to deduce both the set of points to be presented from both images (the union set) and a correspondence between them such that the number of correct mappings is greater than if using only one of the them. Also, a good framework must be able to deal with the salient points that appeared in both of the original correspondences (the intersection set) and points that only appear in one of the correspondences, but not in the other. The scheme of the desired framework is shown in figure 1.3.

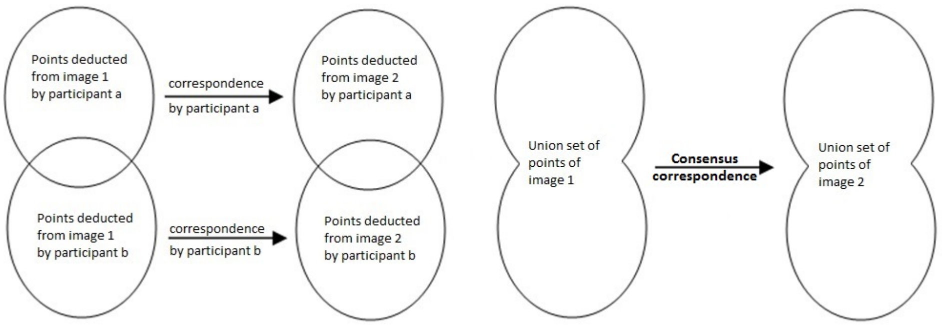

Points deducted from image 1 by participant a
correspondence by participant a
Points deducted from image 2 by participant a
Points deducted from image 1 by participant b
correspondence by participant b
Points deducted from image 2 by participant b
Union set of points of image 1
Consensus correspondence
Union set of points of image 2

Figure 1. 3. Left: Input of the problem. Right: Proposed solution.

1.3 Thesis Structure

This thesis is divided into ten chapters. In the first one, we have presented the most elemental definitions and advantages of using correspondences instead of data structures for pattern recognition purposes, as well as a brief introduction to the issues most commonly found when intending to learn the consensus between correspondences.

Chapter 2 provides the notations and basic definitions required to understand all the methodologies and algorithms to be presented in this work. In essence, this chapter is divided into four subsections. The first one describes the concept of the distance as a basis of the whole work. Then, we develop the fundamental notions of the correspondence generation process in different data structures. Afterwards, some general concepts such as the weighted mean and the median are defined for the current case at hand. Finally, we study the formal concept of consensus frameworks and comment on how it has been used in several areas of knowledge.

In chapter 3, we present a first approach towards learning the consensus of multiple correspondences based on the calculation of the generalised median correspondence. This framework relies on well-known concepts which have been applied previously for the median computation of other data structures.

Chapters 4 through 6 are meant to present in depth the evolution of the work done on the design of a consensus framework for correspondences. This evolution is shown in a sequential order; starting from the basic case of learning a consensus between two correspondences, and finishing with the more challenging task of learning the consensus of multiple correspondences using several variables.

In chapter 7, the consensus framework presented in this work is upgraded through the implementation of an online learning algorithm that automatically deduces the weights for both the correspondences involved in the consensus process and for the restrictions that the model considers.

Chapter 8 presents our newest developments related to the distance and weighted mean search between correspondences. These advances have been presented lately and thus, could not be included in the consensus frameworks discussed in this work. However, we believe it is important that this most recent concepts are well explained in order to give continuity and insights of the future work.

Although we have stated previously that this work is not devoted to present new matching algorithms, but rather on the assumption that correspondences are already given, chapter 9 presents a parallel work in which efforts have been done to define a matching methodology for the specific case of image registration. In this framework, we suppose that one of the images is a tiny patch of the larger one, and a methodology to perform the partial-to-full matching based on multiple candidate centres is described.

Finally, chapter 10 is reserved for conclusions and further work in general terms.

To better understand how all of these information is interconnected in a schematic form, a flow diagram containing all of the aforementioned information is provided in figure 1.4.

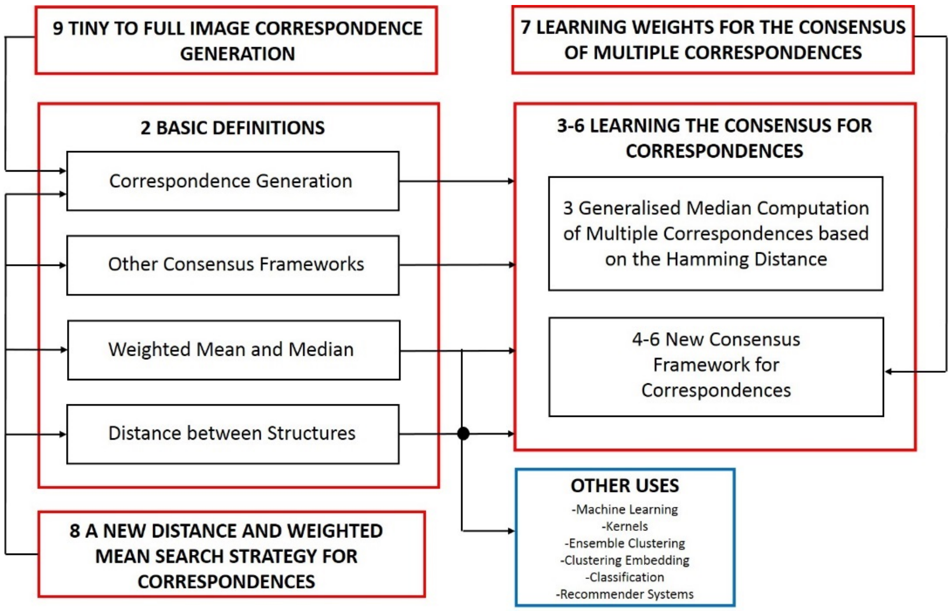

9 TINY TO FULL IMAGE CORRESPONDENCE GENERATION
7 LEARNING WEIGHTS FOR THE CONSENSUS OF MULTIPLE CORRESPONDENCES
2 BASIC DEFINITIONS
Correspondence Generation
Other Consensus Frameworks
Weighted Mean and Median
Distance between Structures
3-6 LEARNING THE CONSENSUS FOR CORRESPONDENCES
3 Generalised Median Computation of Multiple Correspondences based on the Hamming Distance
4-6 New Consensus Framework for Correspondences
OTHER USES
-Machine Learning
-Kernels
-Ensemble Clustering
-Clustering Embedding
-Classification
-Recommender Systems
8 A NEW DISTANCE AND WEIGHTED MEAN SEARCH STRATEGY FOR CORRESPONDENCES

Figure 1.4. Flow diagram representing the structure of the information presented in this work.

Chapter 2 Basic Definition

2.1 The Concept of Distance

A distance in the context of pattern recognition is defined as a similarity function between two objects. It is the most essential concept if a study of data at hand has to be performed. For instance, any application that is designed for matching, classification, kernels, clustering, learning, statistical analysis and structural pattern recognition, relies on the initial establishment of a distance function between the data structures that are being compared.

Several approaches have been presented to establish distance functions in the literature [1], [2]. However, one of the most outstanding advances in pattern recognition has been the introduction of the concept of *edit distance*. This type of distance has been concretised as string edit distance [3] (or Levenshtein distance [4]), tree edit distance [5] and graph edit distance [6], [7], [8], [9] (sets of points can be considered a special case of graphs that do not have edges). In this distance, the dissimilarity between two data structures is defined as the minimum amount of required distortions to transform one of the data structures into the other. To this end, a number of edit operations, consisting of insertion, deletion and substitution of local elements, are established. Edit cost functions are introduced to quantitatively evaluate these edit operations. The basic idea is to assign a penalty cost to each edit operation according to the amount of distortion that it introduces in the transformation. Deletion and insertion operations are transformed to assignments of a non-null element of the first or second structure to a null element of the second or first structure. Substitutions simply indicate the cost of introducing new element-to-element mappings.

2.2 Correspondences and Matching Algorithms

2.2.1 Basic Characteristics of Correspondences

Since the use of correspondences as a data structure *per se* has not yet been documented formally, the following section is devoted to present the essential features that the correspondences generated and managed through the course of this work can present.

2.2.1.1. Definitions for One Correspondence

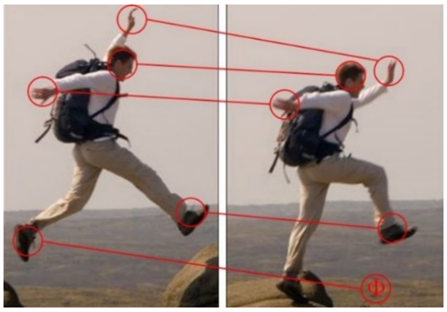
A correspondence f is represented in computer science as an array of numerical references that indicates how the elements of an output object $G = \{a_1, \dots, a_i, \dots, a_N\}$ are mapped with respect to elements of an input object $G' = \{a'_1, \dots, a'_i, \dots, a'_{N'}\}$ in a unidirectional way. Each of these element-to-element relations $f(a_i) = a'_i$ is called a *mapping*. Moreover, mappings could be also interpreted as the unary elements of a larger "data structure" which on this case, would be the correspondence. Following this notion, a correspondence is a set of mappings $f = \{m_1, \dots, m_i, \dots, m_N\}$. Regardless of the interpretation, a correspondence must map the elements of the two objects in an injective form, meaning that all elements of the output element must have a mapping indicated.

Even if $N = N'$, it is not necessary that a correspondence maps both objects in a bijective form. This may occur because there may be no element on the input object that resembles a certain element of the output object. Another factor could be that the matching process considers that one or more elements of either object are spurious. To address this fact, a *null mapping* is used to indicate that an element of the output object has not been mapped. In the array of references, null mappings are marked with a numerical value that distinguishes them from the values used for the elements of the inlier object (i.e. a negative number). More formally, given two objects and a correspondence between them, *inlier elements* are the elements of either the output or the input object that were mapped by the correspondence, and *outlier elements* are the ones that were not.

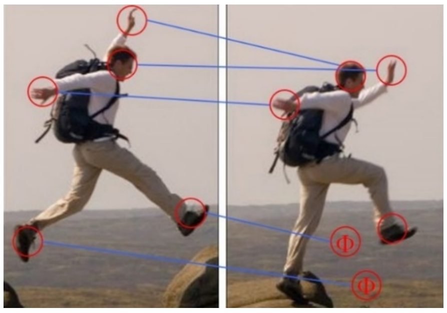
For correspondences to be bijective, after the matching process it is common to extend both objects using *null elements*, which are a supplementary set of elements with features that distinguish them from the rest. As a result, the output and the input objects are extended to be of size $\hat{N}$, and then the correspondence is able to map all elements of the extended output object with all elements of the extended input object. Hence, the correspondence would now be constituted of *inlier mappings* (a mapping between two non-null elements) and *outlier mappings* (a mapping where at least one null element is involved).

2.2.1.2 Definitions for Multiple Correspondences

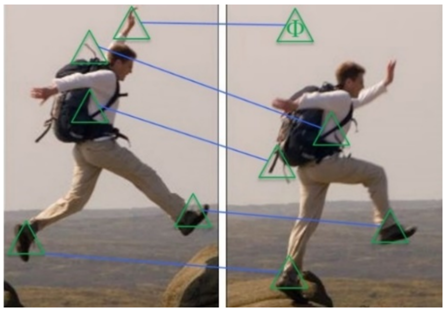
Given two or more correspondences between the elements of two objects, some additional definitions have to be introduced. Figure 2.1 shows three examples of correspondences between two images. These correspondences have been generated through the use of different combinations of feature extractors and matching algorithms. In this example, null elements are identified with Φ.



a) Points detected by feature extractor O and matching algorithm–



b) Points detected by feature extractor O and matching algorithm–



c) Points detected by feature extractor Δ and matching algorithm–

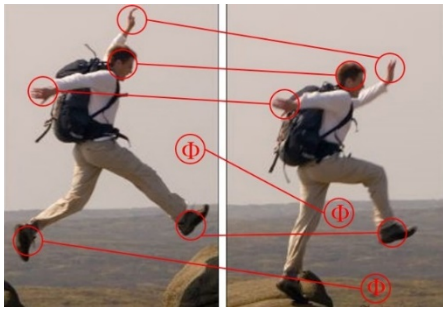
Figure 2. 1. Three correspondences deduced by different feature extractors and matching algorithms.

Two possible scenarios can be identified when using two or more correspondences to learn a consensus. In the first one, the points have been computed by the same feature extractor, and the matching algorithms compute a correspondence from this information (figure 2.1(a) and 2.1(b)). This scenario will be onwards referred as the *joint set scenario*. In the second scenario, the parties seek separately or using interactive methodologies [10] for the points, and then, regardless of the matching algorithm used, the correspondences are distinct (figure 2.1(a) and 2.1(c), figure 2.1(b) and 2.1(c), or all three simultaneously). This scenario will be onwards called the *disjoint set scenario*.

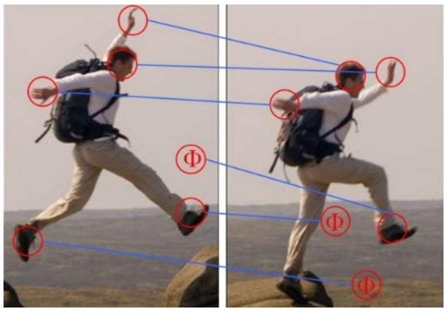
A well-designed framework to learn a consensus must cope with these two scenarios, preferably in an undistinguishable form. Therefore, to convert the disjoint set scenario into the joint set scenario, one option is to use proper intersection functions depending on the information at hand. For instance, methods that rely on the computation of a median graph [11] could be used for this purpose. Also, a method in [12] has been presented to perform a conversion of this nature based on a common labelling. Most recent approaches effectively show that it is possible to make a fusion of graphs that represent the original data, such as the sample graph method [13], the data fusion [14] through the maximum common sub-graph [15], or by merging two sub-graphs to form a larger one base on graph mining techniques [16].

Consider the joint set scenario in the previous example. Notice that the correspondence in figure 2.1(a) is bijective, while the one in figure 2.1(b) is not. By adding a null point in the output image as shown in figure 2.2, it is possible to make the second correspondence bijective, while keeping the first one bijective as well. In addition, both correspondences would map the same points of the output image and the same points of the input image, thus deriving in *mutually bijective* correspondences.

Mutually bijective correspondences present as main advantage that by mapping the same elements, they are better for comparison purposes. Nonetheless, forcing two or more correspondences to be mutually bijective derives in the addition of numerous null elements and outlier mappings, increasing both the size of the objects, the correspondences and the computational demand. As further explanations will show, in some cases it is imperative to work with mutually bijective correspondences, while in some other cases this process can be avoided through the use of complementary definitions.



a) Correspondence in figure 2.1(a) being mutually bijective with respect to correspondence in figure 2.1(b)



b) Correspondence in figure 2.1(b) being mutually bijective with respect to correspondence in figure 2.1(a)

Figure 2. 2. Correspondences in the joint set scenario proposed in figure 2.1 in a mutually bijective form.

2.2.1.3 Distance Between Correspondences

To calculate the distance between two correspondences, the most practical way is through the Hamming distance (HD). Given two correspondences f^1 and f^2, the HD is defined as:

$$Dist_{HD}(f^1, f^2) = \sum_{i=1}^{N} \left(1 - \partial\left(a'_x, a'_y\right)\right) \tag{2.1}$$

being x and y such that $f^1(a_i) = a'_x$ and $f^2(a_i) = a'_y$, and ∂ being the well-known Kronecker Delta function

$$\partial(x, y) = \begin{cases} 0 \ \ if \ x \neq y \\ 1 \ if \ x = y \end{cases} \tag{2.2}$$

The larger the value of the HD, the more dissimilar two correspondences are. Moreover, we consider the following rules for the HD calculation given the correspondence definitions presented above:

1. Both correspondences must map every element of the output object, and therefore be of the same size.
2. Both correspondences must be part of a joint set scenario.
3. If correspondences are mutually bijective and have been extended with one or multiple null elements in the output object, then the HD should not take into consideration the mappings of the null elements of the output object.
4. If the correspondences are mutually bijective and have been extended with more than one null element in the input object, then the HD should take into consideration that if $f^1(a_i) = a'_x$ and $f^2(a_i) = a'_y$, but a'_x and a'_y are null elements, then the distance value must not increase.

The HD has proven to be a reliable choice to compute the distance between correspondences, given the fact that if the only interest is to establish a dissimilarity function between the information provided by the correspondences alone, a comparison of the mappings is sufficient, with the great benefit of a minimal computational time to obtain the distance. Nonetheless, and as will be shown throughout this work, we acknowledge that there is a requirement of a more robust distance between correspondences, especially if additional characteristics (i.e. structural information of the objects matched) has to be considered to express the distance.

2.2.2 Correspondence Generation

In this section, the particularities of performing the matching process different data structures will be discussed, emphasising on salient point correspondences for non-structural pattern recognition, and graph correspondences for structural pattern recognition. As commented in chapter 1, one of the main motivations of the present work is to learn the consensus of correspondences instead of the data structures being mapped and therefore, the objective of this work is not to present matching algorithms (although a matching framework for a specific scenario is presented in chapter 9). Nonetheless, we believe it is important to identify how correspondences can be generated and the different information contained depending on the type of object mapped.

2.2.2.1. Minimum Cost Matching

One of the basis of matching algorithms, regardless of the data structure used, is to produce a correspondence which obtains the minimum cost, that is, the distance between the attributes or features of the objects being matched. To that aim, the cost of a correspondence $Cost(G, G', f)$ is defined as the addition of the cost of each local element in a similar way as in the edit distance.

$$Cost\left(G, G^{'}, f\right) = \sum_{i=1}^{N} c\big(a_i\, , f(a_i)\big) \tag{2.3}$$

where c is a distance function over the domain of attributes of the elements being mapped. Moreover, distance c is application dependent, and has to consider both the case where two original elements are mapped, regardless if they are inlier elements or null elements ($c = 0$ if the mapped elements are null elements).

The most practical way to calculate the cost of a correspondence $\boldsymbol{C}_f$ is through the use of a cost matrix $\boldsymbol{C}$. This way, we convert the minimisation problem into an assignment problem [17]. First, a cost matrix is constructed using the cost values of every possible combination between all elements of G and G'. Then, a cost $\boldsymbol{C}_f$ is obtianed as the addition of every possible mapping:

$$\boldsymbol{C}_f = \sum_{i=1}^{N} \boldsymbol{C}[i, k] \; where \; f(a_i) = a'_k$$

$$s.t. \forall i \neq j, f(a_i) \neq f(a_j) \tag{2.4}$$

As a result, it is safe to say that the following expression is true whenever a cost calculation is performed:

$$Cost\left(G, G^{'}, f\right) = \boldsymbol{C}_f \tag{2.5}$$

The correspondence that obtains the minimum distance is known as the optimal correspondence f^*, and can be obtained through the minimisation of the cost matrix. This can be done by applying a linear assignment problem

(LAP) solver such as the Hungarian method [18], the Munkres' Algorithm [19] or the Jonker-Volgenant solver [20].

2.2.2.2 Salient Point Correspondences

Salient points are one of the simplest data structures used to generate a correspondence, since matching algorithms consider only the attributes of each point. Correspondences of this kind are typically used for image registration, which is the process of transforming different sets of images into one coordinate system of points. These points can be corners (intersection of two lines in the image) [21] maximum curvature points [22] or isolated points of maximum or minimum local intensity [23]. There is an evaluation of the most competent approaches for feature extraction in [24]. Salient point correspondences are used in computer vision, medical imaging, data obtained from satellites and image analysis in general, by integrating data obtained from multiple sources; a process commonly referred as image registration. Interesting image registration surveys are [25], [26] and [27], which explain the problematic of this goal.

Image registration methods are usually composed of two steps [25]. First, some salient points are selected from both images by using a feature extractor. Several methods have appeared to select salient points in images [28], most notably the following:

1. FAST [29]: It is composed of a vector of features obtained from an algorithm called "Accelerated Segment Test". This extractor uses an approximation metric to determine the corners of an image.
2. HARRIS [30]: It is composed of a vector of features obtained from the Harris-Stephens algorithm. It is able to find corners and edges based on a local auto-correlation function.
3. MINEIGEN [31]: It is composed of a vector of features obtained from the minimum eigenvalue algorithm. This algorithm determines the location of the corners based on the eigenvector and eigenvalue domain. It is originally designed for tracking purposes and it is able to detect some occlusions in the image.
4. SURF [32]: It is composed of a vector of features obtained from the "Speeded-up Robust Features" algorithm. It is able to detect multiscale objects (known as blobs), as well as scale and rotation changes.
5. SIFT [33]: It is composed of a vector of features obtained from the "Scale-Invariant Feature Transform" algorithm. It applies the Gaussian

difference given several regions of the image in order to find scale and rotation features.

The aforementioned methods are based on assigning some local features (for instance, a vector of 64 features) to each extracted point or pixel of the image. Each local feature usually depends on the information of the image.

The second step is based on finding a correspondence between the extracted salient points. This goal can be achieved either by using a matching algorithm between the two sets of points, or by using algorithms which aim at transforming the coordinate system of one of the images into the other one and thus, deduce an image homography. If the objective is only to obtain a correspondence, then a minimum cost matching approach would be sufficient for this purpose. Conversely, if the interest is to also obtain the image homography, other approaches must be used instead. For instance, the Iterative Closest Point (ICP) [34] algorithm can be employed to minimize the difference between two clouds of points. ICP is often used to reconstruct 2D or 3D surfaces from different scans. It only uses the position of the points, but not the local features. Nevertheless, it has the advantage of obtaining not only an homography, but a correspondence as well. It is typical to use ICP together with RANSAC [35], which is a method that discards points that do not fit on the deduced homography and correspondence and so, eliminates spurious mappings. That is, the algorithm looks for salient points that have been considered to appear due to noise in the images. Also, the Hough transform [36], [37], [38] is a classical technique used to find imperfect instances of objects represented by sub-sets of salient points within an image by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. More recently, an algorithm [39] which consider the features of each point and also the homographies, has been proposed for salient point matching.

Several frameworks that intend to solve the whole image registration problem can be found in the literature. These approaches consider the whole set of salient points of each image, which do not necessarily represent a unique system having the same coordinates or characteristics for both parts. That is, these methods take into consideration that the salient points are grouped in several sets of points, since they assume different transformations or homographies are applied to each point set [40]. In [41], authors explicitly considered some different levels of occlusion and noise in the object's contour. To enhance the quality of the correspondence, some methods have been presented which register several images at a time to increase the probability of finding successful matches [42], or that use vector field learning to refine correspondences, removing spurious or mismatch mappings [43].

2.2.2.3 Palmprint Image Registration

2.2.2.3.1 State of the Art Methods

One particular case of study in matching applied for image recognition is related to palmprint matching. The use of palmprint recognition has increased in recent years with respect to fingerprint recognition in forensics applications [44], since palmprints provide a larger area to extract information and thus, to identify a person. In palmprint image registration, characteristics of the images and the features obtained from them lead to particular methodologies.

While the general image registration applications use the aforementioned feature extractors to detect and extract image features, in the case of palmprint, the points to detect are called minutiae, and the features extracted from them are different than the standard feature vector. The typical features extracted from minutiae are the location (x, y coordinate), the type (bifurcation, termination or dot) and the directional angle of the ridge line in which they are located. For illustrative reasons, figure 2.3 presents an example of the ridge lines of a palmprint and the types of minutiae that can be found in them.

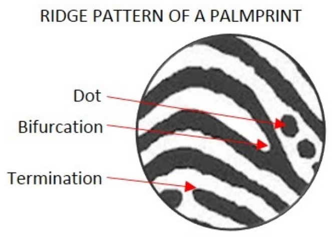


Figure 2. 3. Sample portion of a palmprint ridge pattern.

Several methods have been presented for palmprint image matching throughout the course of recent years [45], [46], [47], [48]. In particular, an approach modelled by [49] proposed a low resolution image matching based on the palmprint ridges. Also, [50] presented a novel algorithm for minutiae matching using crease elimination. More recent approaches propose different matching criteria than merely minutiae analysis. Inspired in the work of [51] and [52] for latent (found in scene) palmprint matching, a document presented by [53] proposed a multi-feature fusion algorithm that showed an improvement in matching percentage with respect to its predecessors. Nonetheless, multi-feature extraction depends on the image quality. In forensic sciences, the quality of latent palmprints tends to be low and thus, methods of this nature have little use. Later, [54] proposed a robust ridge-based matching algorithm which outperforms [53]. However, this method is once again dependent of more features beside than the ones from minutiae. A method based on wavelets has been more recently presented in [55] using only ridge-based information. Also, a method presented in [56] obtains a fingerprint distance that can also be applied for palmprint matching, but that needs several samples of the same palmprint since it is based on the Principal Component Analysis (PCA) [57]. Later in a method presented by [58], the PCA is also used, but each palmprint is divided into several square-overlapping blocks. This was done to classify these blocks into either a good block or a non-palmprint block. Finally, a method was presented in [59], where palmprint images are decomposed by 2D Gabor wavelets. The drawback of this methodology is that this decomposition is said to be sensitive to the length of the obtained palmprint.

As commented in chapter 1, one additional contribution of this work is to present an image registration framework where the output image is much smaller than the input one. This methodology is inspired by our ongoing study of palmprint matching, and will be described in depth in chapter 9. Therefore, further definitions of palmprint matching as well as more methods specifically applied in this scenario will be provided further in this work.

2.2.2.3.2 Minimum Cost Minutiae Matching

To obtain a correspondence between output palmprint P and input palmprint P', a cost c_{mnu} between minutiae must be defined first. Consider two minutiae m_i^P and $m_j^{P'}$ extracted from palmprints P and P' can be represented by positions $\left(x_i^P, y_i^P\right)$ and $\left(x_j^{P'}, y_j^{P'}\right)$, as well as two sets of features $\gamma_i^P = \left(\theta_i^P, t_i^P\right)$ and $\gamma_j^{P'} = \left(\theta_j^{P'}, t_j^{P'}\right)$. Feature θ_i represents the directional angle of the ridge where the minutia is located, and t_i represents the type of minutia. If $t_i^P \neq t_j^{P'}$ then both minutiae are not the same type and

$c_{minutiae}\left(m_i^P, m_j^{P'}\right) = \infty$. Otherwise, both an angular cost and a position cost can be obtained.

The angular cost is defined as

$$c_{ang}\left(f_i^P, f_j^{P'}\right) = cyclical\_dist\left(\theta'^P_i, \theta'^{P'}_j\right) \tag{2.6}$$

where angles θ'^P_i and $\theta'^{F_a}_j$ are the normalised angles of the minutiae with respect to the average angle, that is $\theta'^P_i = \theta_i^P - \bar{\theta}^P$ and $\theta'^{P'}_i = \theta_i^{P'} - \bar{\theta}^{P'}$. This is done to have a rotation invariant matching. Moreover, the position cost is defined as

$$c_{pos}\left(\left(x_i^P, y_i^P\right), \left(x_j^{P'}, y_j^{P'}\right)\right) = Euclidean\left(\left(x'^P_i, y'^P_i\right), \left(x_j^{P'}, y_j^{P'}\right)\right) \tag{2.7}$$

where $\left(x'^P_i, y'^P_i\right)$ is the position of minutiae $\left(x_i^P, y_i^P\right)$ to which a translation towards the position of the candidate centre $(\bar{x}^{P'}, \bar{y}^{P'})$, and also a rotation, have been applied. The angle for this rotation is derived from the difference between the average angles calculated for the angular cost, that is $\bar{\theta}^{rot} = |\bar{\theta}^P - \bar{\theta}^{P'}|$. This results in

$$\left(x'^P_i, y'^P_i\right) = rotate_{\bar{\theta}^{rot}}\left(\left(\left(x_i^P, y_i^P\right) - (\bar{x}^P, \bar{y}^P)\right) + (\bar{x}^{P'}, \bar{y}^{P'})\right) \tag{2.8}$$

With these two costs, we define the minutiae cost c_{mnu} between m_i^P and $m_j^{P'}$ as

$$\begin{aligned} c_{mnu}\left(m_i^P, m_j^{P'}\right) = \lambda_c \cdot c_{ang}\left(f_i^P, f_j^{P'}\right) + (1 - \lambda_c) \\ \cdot c_{pos}\left(\left(x_i^P, y_i^P\right), \left(x_j^{P'}, y_j^{P'}\right)\right) \end{aligned} \tag{2.9}$$

where weight λ_c depends on the data at hand.

2.2.2.4 Graph Correspondences

Graph matching is one of the most widely used forms of pattern recognition. Several state of the art graph matching algorithms [60], [61] have been presented in the last 40 years, with surveys such as [62], [63] and [64] exploring the most representative proposals. From this vast pool of options, the graph edit distance [6], [7], [8], [9] appears as one of the most interesting concepts to define not only a distance between graphs, but also a node-to-node or an edge-to-edge correspondence.

Given two graphs G and G', and a correspondence f between their nodes and/or edges, the edit cost between the graphs is:

$$EditCost(G,G',f) = \sum_{\substack{v_i \in \Sigma-\hat{\Sigma} \\ v'_j \in \Sigma'-\hat{\Sigma}'}} d(v_i, v'_j) + \sum_{\substack{v_i \in \Sigma-\hat{\Sigma} \\ v'_j \in \hat{\Sigma}'}} K + \sum_{\substack{v_i \in \hat{\Sigma} \\ v'_j \in \Sigma'-\hat{\Sigma}'}} K \tag{2.10}$$

where $f(v_i) = v'_j$, d is a distance between elements, and constant K is the penalty of deleting or inserting elements. Then, the graph edit distance between the graphs is the minimum cost under any bijection in the set of all correspondences T:

$$EditDist(G,G') = \min_{f \in T}\{EditCost(G,G',f)\} \tag{2.11}$$

Since its inception, important contributions to understand and implement the graph edit distance have been presented [65], [66]. However, none of them has been more crucial than the Bipartite (BP) graph matching framework [67]. This algorithm considers the local features of each element, and is based on converting the LAP into a correspondence problem with the addition of offering spurious mapping rejection through the definition of insertion and deletion costs.

The BP framework is composed of three main steps. The first step defines a cost matrix, the second step applies a LAP solver such as the Hungarian method [18], the Munkres' algorithm [19] or the Jonker-Volgenant solver [20] to this matrix and obtains the correspondence f^*, and the third step computes the edit distance cost given this correspondence between both objects, that is $EditDist(G,G') = Cost(G,G',f^*)$.

More specifically, given two graphs G and G', and a correspondence f between them, the graph edit cost, represented by the expression $Cost(G,G',f^*)$, is the cost of the edit operations that the correspondence imposes. It is based on adding the following constants and functions:

1. C_{vs} is a function that represents the cost of substituting node v_i of G by node $f(v_j)$ of G'.

2. C_{es} is a function that represents the cost of substituting edge $e_{i,j}$ of G by edge $f(e_{i,j})$ of G'.
3. Constant K_v is the cost of deleting node v_i of G (mapping it to a null node) or inserting node v_j' of G' (being mapped from a null node).
4. Constant K_e is the cost of assigning edge $e_{i,j}$ of G to a null edge of G', or assigning edge $e_{i,j}'$ of G' to a null edge of G.

For the cases in which two null nodes or two null edges are mapped, the yielded cost is zero. Figure 2.4 shows the cost matrix used on the BP algorithm.

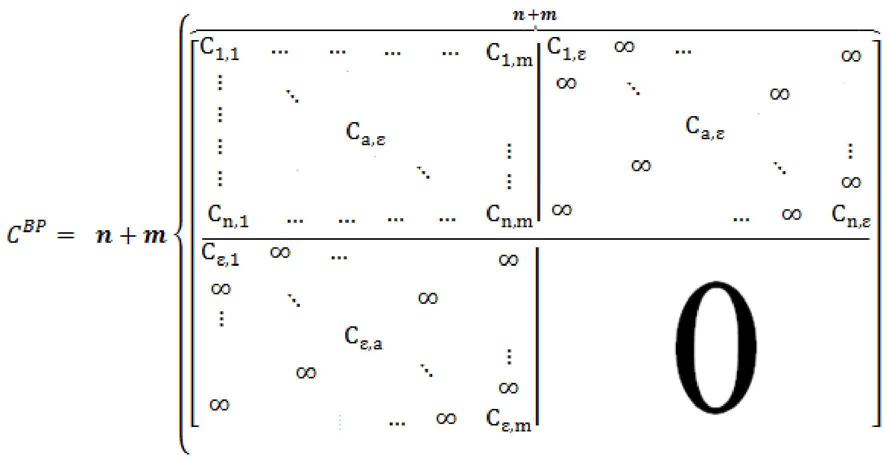


Figure 2. 4. Cost matrix of the BP algorithm.

The upper-left quadrant is composed of substitution costs $C_{i,j}$, which represent the whole cost of substituting v_i for v_j considering their local sub-structure. The diagonal of the upper-right quadrant denotes the whole costs $C_{i,\varepsilon}$ of deleting nodes v_i and their local sub-structure. Similarly, the diagonal of the bottom-left quadrant denotes the whole costs $C_{\varepsilon,j}$ of inserting node v_j' and its local sub-structures. Finally, the bottom-right quadrant is filled with zero values, since the substitution between null nodes has a zero cost. A couple of more recent versions of the BP framework, called the Fast Bipartite (FBP) [68] and the Square Bipartite (SFBP) [69], propose some improvements on this matrix to reduce the computational time and improve the accuracy. proposals

In this work, whenever a graph correspondence has to be generated through the BP framework or has to be used in combination with other graph correspondences to learn the consensus, we will rely on the most common local sub-structures used in graph matching [70], [71], viz. Node, Degree and Clique. In Node, edges are not considered at all. This means that the costs depend on only the node substitution, insertion and deletion costs. In Degree, costs are composed of the node plus the set of neighbouring edges. Finally, in Clique costs are composed of the node, the set of neighbouring edges and also the set neighbouring nodes.

2.3 Weighted Mean and Median

Useful concepts such as the weighted mean and the median are important when intending to bridge the gap between statistical and structural pattern recognition [72], that is, to incorporate well-established techniques from the statistical domain to perform structural tasks. In addition, in this work these concepts also come in great use for the frameworks that intend to learn the consensus of correspondences. Therefore, in the next subsections the definitions of the weighted mean and the median for correspondences are presented.

2.3.1 Weighted Mean

Given two objects e^1 and e^2 and a distance $Dist(e^1, e^2)$, the weighted mean is any object found in between such distance:

$$\begin{gathered} Dist(e^1,\ \bar{e}_w) = \alpha_w \\ \text{and } Dist(e^1, e^2) = \alpha_w +\ Dist(\ \bar{e}_w, e^2) \end{gathered} \tag{2.12}$$

where α_w is a constant that cont7ols the contribution of each object, and must hold $0 \leq \alpha_w \leq Dist(e^1, e^2)$. Note that objects e^1 and e^2 hold this condition and therefore, are also weighted means. Moreover, notice that if $\alpha_w = \frac{1}{2}$ then the exact mean of the two objects has been found. The concept of weighted mean has been defined in literature for data structures such as strings [73], graphs [74] and data clusters [75].

For the case of correspondences, the weighted mean $\bar{f}_w$ is a correspondence that holds,

$$\begin{gathered} Dist\big(f^1, \bar{f}_w\big) = \alpha_w \\ Dist(f^1, f^2) = \alpha_w + Dist\big(\bar{f}_w, f^2\big) \end{gathered} \tag{2.13}$$

Whenever a correspondence fulfils equation 2.13, from now on this fact will be referred as accomplishing the *weighted mean condition*.

It is interesting to observe that in the case of correspondences; any correspondence is a weighted mean if it holds $\bar{f}_w(a_x) = f^1(a_x)$ or $\bar{f}_w(a_x) = f^2(a_x)$ for all the mappings of non-null elements a_x. This property facilitates the search for weighted mean correspondences, as will be shown throughout this work.

Usually, a weighted mean correspondence given a specific α_w is not unique. This is the reason why a search strategy to compute the weighted means of a pair of correspondences must be defined in order to use this concept for learning the consensus. For data structures such as strings, graphs and data clusters, authors have relied on computing the optimal set of edit operations between the first and second data structure, and then applying a subset of these edit operations to the first element to produce a new element. This new element is guaranteed to be a weighted mean. Most notably in the work presented for data clusters [75], authors formalise definitions and present two methods to execute the weighted mean search: *Moving Elements Uniformly* and *Moving Elements by using Cluster Means*.

To find weighted mean correspondences in an intuitive form, a first proposal for a weighted mean search strategy was developed based on *Moving Elements Uniformly*. Given two mutually bijective correspondences f^1 and f^2 and the HD as the distance between them, this proposal initially defines one of these correspondences as the weighted mean correspondence $\bar{f}_w$. Then, it gradually swaps pairs of mappings in $\bar{f}_w$. Notice that whenever a swap is performed, the values $Dist_{HD}(f^1, \bar{f}_w)$ and $Dist_{HD}(\bar{f}_w, f^2)$ may change, but not necessarily at the same rate. For this reason, the algorithm checks after each swap if the current $\bar{f}_w$ holds the weighted mean condition. If it does, a weighted mean has been formed and the swapping continues on this newly formed weighted mean correspondence until finding all possible weighted means (that is, when $\bar{f}_w = f^2$). If it does not, the process is reset and repeated. Notice that for this process to be properly executed, both correspondences must be mutually bijective. From now on, this weighted mean search strategy will be referred as *Moving Mappings Uniformly*.

2.3.3 Median

Given a set of M objects, the median object $\hat{e}$ is defined as the one that has the smallest sum of distances to all correspondences in the set [76].

$$\hat{e} = \underset{\forall e}{argmin} \sum_{i=1}^{M} Dist\left(e, e^i\right) \tag{2.14}$$

From this definition, we are able to discern two concepts: the generalised median (GM) and the set median (SM). The difference lies in the space where each median is searched. In the first case, there are no restrictions (the median may be any object in the whole domain of possible objects) while in the second case, the exploration space is restricted to the M objects in the set. It is interesting to notice that in the case of the median between two objects, it has been proven in [77] that any weighted mean is also the median of such two objects, since the distance to the two objects is minimised equally by any of the weighted means.

The concept of the GM of data structures has been used in the last years for applications such as learning [78], shape recognition [79], *k*-means clustering [80] and most notably clustering ensemble [81], [82], since it is a form of searching for a representative prototype of a set of objects. In contrast to the generalised mean [83], [84], the GM finds this representative prototype with a resilience to outlier objects. However, for most of these cases, the GM computation turns out to be an NP-complete problem. This drawback has led to a variety of methods solely developed for the GM approximation, such as a genetic search [11], [76], lower bound estimations [85] and iterative approximations for strings based on edit operations [86]. Other approaches are based on the weighted mean calculation, such as the embedding methods have been applied on strings [87], graphs [88], [89] and data clusters [90]. Recently, a method known as the evolutionary method [91] has been presented, which offers a good trade-off between accuracy and runtime, making it one of the most viable option for the GM approximation on correspondences. In the next chapter, the definitions and a framework to calculate the GM correspondence based on the HD will be presented.

2.4 Consensus Frameworks

As noticed, the concepts defined so far aid to learn a consensus of multiple correspondences. To complement them, several methodologies have been studied, which also intend this goal, but considering diverse factors. These methodologies are called *consensus frameworks*; a concept that has been previously proposed both for matching purposes [92] and for areas that diverge from classical pattern recognition.

Whilst classical minutiae matching used for fingerprint and palmprint identification comprises most of the literature, methods of this kind present two major flaws. Firstly, minutiae are hard to extract from low image resolution or latent samples. Secondly, these points may not be necessarily the most significant data from a specific region of the finger or the palm. Therefore, there has been interest in developing consensus frameworks for fingerprint and palmprint matching [93], [94], [95], which rely on combining the aforementioned minutiae-based approaches with less conventional matching algorithms based on other features such as the orientation field [96], as well as the shape [97] and density [98] of the ridges. These frameworks first perform a consensus of the features, to then produce a so called consensus matching function for such features.

Several other examples of consensus frameworks can be found in alternate literature. For instance, voting systems are the most recurrent tool to learn a consensus. The concept of voting has been long studied in various fields, and is essentially devoted to construct a final representation based on the opinions of the different involved parties. Such opinions are gauged by a set of predefined rules, being the most common one the majority voting rule, although numerous alternative criteria have been presented as well [99]. Besides their logical uses in politics and economics, voting methodologies have been developed in systems more related to computer sciences. Some examples are [100] and [101] for multiple classifier combination, [102] for creating more robust point matching algorithms and [103], [104] for confidence voting methods. Furthermore, voting methodologies go hand by hand with clustering ensemble methodologies, as shown in [105] and [106].

Collaborative methods are other proposals that are constantly related to consensus frameworks. In them, several entities provide resources for the announcement of a final result, but do not necessarily cooperate in the same form as in voting schemes. Recently, a collaborative method has been proposed where, given a set of classifiers, return the most promising class [107]. This method, based on a previous development for face recognition [108], learns some weights that gauge the importance of each classifier and also the sample through several techniques such as voting strategies or hierarchical structures. Also, collaborative frameworks have been adapted to machine learning [109] and filtering [110], which has found a valuable use in content-based filtering [111] for recommender systems [112].

Another example is consensus clustering, which is a particular form of dealing with the clustering problem of unlabelled data applying unsupervised learning methodologies [113]. Method to perform consensus clustering are [114], where a final cluster is obtained based on a similarity graph, or [115], where the least square algorithm is used. Moreover, in [116] consensus clustering is used to classify genetic diseases.

Finally, there are consensus frameworks proposed for decision making in scenarios different from conventional pattern recognition. Interesting papers are [117], a trust consensus is achieved given a multi-person analysis, [118], two metrics are used for social network analysis, [119] in which active learning for webpage classification is enhanced using a multiple-view multiple-learner approach, and [120], where a consensus decision is taken given some decisions between a group of people applying fuzzy logic techniques. Moreover, it is worth to mention methods such as [121], where a consensus framework was designed to model non-linear utility constraints and [122], where authors define exit-delegation mechanisms in case of emergencies.

Chapter 3 Generalised Median Computation of Multiple Correspondences based on the Hamming Distance

3.1 Introduction

The computation of a generalised median (GM) is typically an *NP*-complete task. Nonetheless, it has been proven to be a very reliable form to compute a representative prototype for most of the data structures studied in pattern recognition and computer vision, such as sets of salient points, strings, trees, graphs and data clusters. Therefore, the GM could be used as a first option to learn the consensus of correspondences. In this chapter, we present two methodologies to obtain the GM of multiple correspondences. The first one is based on a voting process solved through a minimisation process. This solution is proven to obtain the optimal solution in cubic time, but is hold to some restrictions. The second one is based on the evolutionary method presented in [1] adapted to the correspondence case. This solution obtains a suboptimal solution through an iterative approach, but does not have any restrictions with respect to the used distance.

3.2 Definition and Notations

Given two objects $G = (\Sigma, \Upsilon)$ and $G' = (\Sigma', \Upsilon')$ of the same order N (naturally or due to a null element extension), we define the set of all mutually bijective correspondences $T_{G,G'}$ such that each correspondence in this set maps elements of G to elements of G', $f: \Sigma \to \Sigma'$ in bijective manner. Onwards, we refer to T instead of $T_{G,G'}$. Moreover, consider a subset of correspondences $S \in T$ between the same pair of objects that have been produced using different matching approaches. Given M input correspondences, the GM correspondence is the correspondence in T that has the smallest sum of distances to all correspondences in such set.

$$\hat{f} = \underset{\forall f \in T}{\operatorname{argmin}} \sum_{i=1}^{M} Dist(f, f^i) \tag{3.1}$$

Notice the calculation of the GM is closely related to the distance between the objects involved, and thus, the importance of defining that the Hamming distance (HD) will be used for the current framework.

3.3 Methodology

In this section, two methods to calculate the GM are presented, onwards referred as *minimisation method* and *evolutionary method*.

3.3.1 Minimisation Method

The name of this method is due to the minimisation of the sum of the linear assignment problem (LAP). To better explain it, the following practical example is introduced. Suppose that three separate entities have proposed mutually bijective correspondences as shown in figure 3.1, depicted as f^1 (red lines), f^2 (blue lines) and f^3 (green lines).

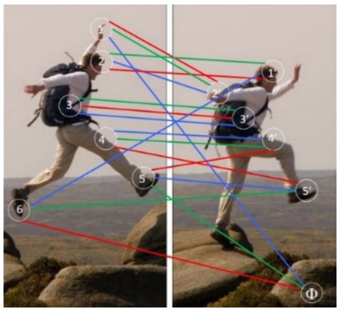
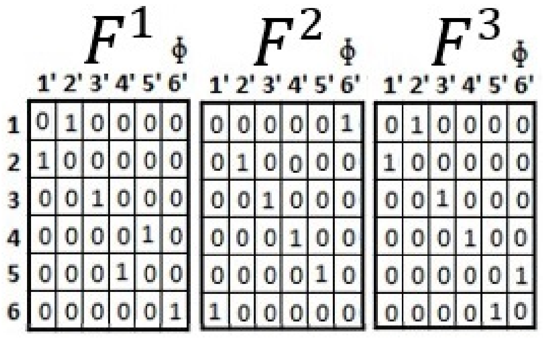


Figure 3. 1. Practical example of three correspondences.

It is possible to represent these correspondences through the use of correspondence matrices F^1, F^2 and F^3 as shown in figure 3.2. These matrices are defined as $F^k[x,y] = 1$ if $f^k(v_x) = v'_y$, and $F^k[x,y] = 0$ otherwise.



$$F^1 \quad \begin{array}{c|cccccc} & 1' & 2' & 3' & 4' & 5' & \overset{\phi}{6'} \\ \hline 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \qquad F^2 \quad \begin{array}{|cccccc} 1' & 2' & 3' & 4' & 5' & \overset{\phi}{6'} \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{array} \qquad F^3 \quad \begin{array}{|cccccc} 1' & 2' & 3' & 4' & 5' & \overset{\phi}{6'} \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Figure 3.2. Correspondences matrices F^1, F^2 and F^3.

The method proposed in this section minimises the following expression:

$$\hat{f} = \underset{\forall f \in T}{\operatorname{argmin}} \left\{ \sum_{x,y=1}^{N} [H \circ F]\{x,y\} \right\} \tag{3.2}$$

where $\{x,y\}$ is a specific cell and H is

$$H = \sum_{k=1}^{M} \mathbf{1} - F^k \tag{3.3}$$

with $\mathbf{1}$ being a matrix of all ones, F being the correspondence matrix of $f \in T$ (if $f(v_x) = v'_y$ then $F\{x,y\} = 1$, and $F\{x,y\} = 0$ otherwise), and $\circ$ being the Hadamard product.

The method deduces $\hat{f}$ through equation 3.2, and since it is intended to minimise the sum of distances of all correspondences (equation 3.1), then by definition it would deliver the exact GM. This is demonstrated by showing that the obtained correspondence $\hat{f}$ in equation 3.1 is the same than the one in equation 3.2. For this reason, the following equality must hold:

$$\sum_{k=1}^{M} Dist_{HD}(f, f^k) = \sum_{x,y=1}^{N} \left[\left[\sum_{k=1}^{M} \mathbf{1} - F^k \right] \circ F \right] \{x,y\} \tag{3.4}$$

Applying the associative property of the Hadamard product, the following function is obtained:

$$\sum_{k=1}^{M} Dist_{HD}(f, f^k) = \sum_{k=1}^{M} \left(\sum_{x,y=1}^{N} \left[[\mathbf{1} - F^k] \circ F \right] \{x,y\} \right) \tag{3.5}$$

If it is demonstrated that each individual term holds the equality $Dist_{HD}(f, f^k) = \sum_{x,y=1}^{N} \left[[\mathbf{1} - F^k] \circ F \right] \{x,y\}$, then for sure equation 3.4 holds. By definition, the HD counts the number of mappings that are different between the two correspondences and similarly, expression $\sum_{x,y=1}^{N} \left[[\mathbf{1} - F^k] \circ F \right] \{x,y\}$ counts the number of times $F\{x,y\} = 1$ at the same time that $F^k\{x,y\} = 0$. Therefore, the equality is confirmed for the cases where all mappings are counted (that is, mappings of inlier elements on the output object). ■

Notice that by adding all correspondence matrices in equation 3.3, a structure similar to a voting matrix [2] is created. This resulting matrix can be minimised by solving the LAP [3] using either the Hungarian method [4], the Munkres' algorithm [5] or the Jonker-Volgenant solver [6] as shown in algorithm 3.1.

Algorithm 3.1 Minimisation Method

```
Input: A set of correspondences
Output: GM correspondence $\hat{f}$
Begin
1. Compute matrix $H$ (equation 3.3)
2. $\hat{f} = linear\ solver\ (H)$
End algorithm
```

Since LAP solvers have all been demonstrated to optimally minimise a value [3] and equation 3.4 holds, then the method presented in this section obtains the exact GM. Moreover, the method considers only first order information, and no second order information is introduced to the model. That is, the method is not capable of taking into account the attributes or the interactions within the elements of the data structures mapped (for instance, the structural information of nodes in a graph). Therefore, this method is limited to the use of mapping-to-mapping distances such as the HD. Figure 3.3 shows the GM correspondence obtained in the practical example.

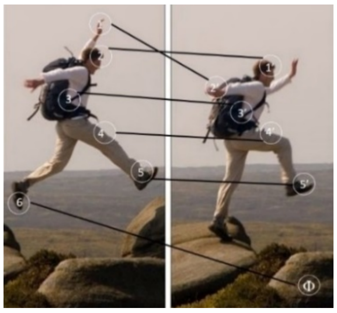


Figure 3.3. GM correspondence of the three correspondences proposed in the practical example

3.3.2 Evolutionary Method for Correspondences

This proposal relies on the GM approximation through the use of weighted means. As proven in [7], the GM of a set of objects in any space can be estimated through an optimal partition of the pairs of weighted mean objects. This is because it is demonstrated that by computing the weighted mean of such optimal pairs of objects, all of those weighted means tend to match in one object that can be considered a good estimation of the GM of the set. Since in some cases the GM can be far away from the deduced object, the iterative algorithm proposed in [1] tends to achieve the true GM. This algorithm, applied to the correspondence domain consists of the following steps,

Algorithm 3. 2 Evolutionary Method

```
Input: A set of correspondences
Output: GM correspondence $\hat{f}$
While convergence
    1. Deduce the optimal pairs of correspondences in the set.
    2. Estimate $\Omega$ weighted means per each pair.
    3. Add the weighted means to the current set.
    4. Find the optimal correspondence(s) in the current set (lowest SOD).
end while
End algorithm
```

The following subsections detail steps 1, 2 and 4 for the correspondence domain. Notice the third step is simply adding the obtained weighted mean correspondences to the current set of correspondences, and thus no further description is required.

3.3.2.1 Optimal Pairs of Correspondences

The first step is to generate the distance matrix given the whole correspondences using the HD. Then, the optimal pairs of correspondences are the ones that produce the minimum SOD between them [7]. Thus, it is only necessary to obtain the pairs of correspondences by applying a LAP solver such as the Hungarian method [4], the Munkres' algorithm [5] or the Jonker-Volgenant solver [6]. Note that one correspondence must not be assigned as the optimal pair of itself and for this reason, instead of filling the diagonal of the distance matrix with zeros as done typically in methods of this nature, high values are imposed instead. Nevertheless, if there is an odd number of correspondences, for sure the solver returns a correspondence mapped to itself. In this case, this correspondence is stored until step 3.

3.3.2.2 Selecting Weighted Means of Pairs of Correspondences

The aim of the step 2 is to estimate Ω equidistant weighted means for each of the pairs of correspondences found in the previous step. Thus, we generate $\bar{f}_{\alpha_1}, \ldots, \bar{f}_{\alpha_\Omega}$ such that $\alpha_i = \frac{i}{\Omega+1}$. The order of Ω is usually set from 1 to 3. This is because, through the practical validation, it has been observed that restricting the process to calculate only the mean correspondence (that is $\Omega = 1$) makes the process converge slower than when having 2 or 3 equidistant weighted means instead, even if such means are obtained in a suboptimal form. Moreover, experimentation has shown that if $\Omega > 3$, the computational cost is increased without gaining in accuracy. We use the *Moving Mappings Uniformly* strategy presented in chapter 2 to perform this weighted mean search.

3.3.2.3 Selecting the Optimal Correspondence

Once the current correspondences are put together with the new weighted mean correspondences to enlarge the set (step 3), the method could return to step 1 with this newly enlarged set, without executing step 4. Nevertheless, the computational cost and memory space needed in each iteration would exponentially increase. For this reason, the aim of this step is to discard the correspondences that are believed not to be a good choice for the GM. To that aim, a distance matrix is computed between the whole correspondences through the use of the HD. Then, the ones that have a larger SOD from themselves to the rest are discarded. Note that this methodology is in line of the definition of the GM

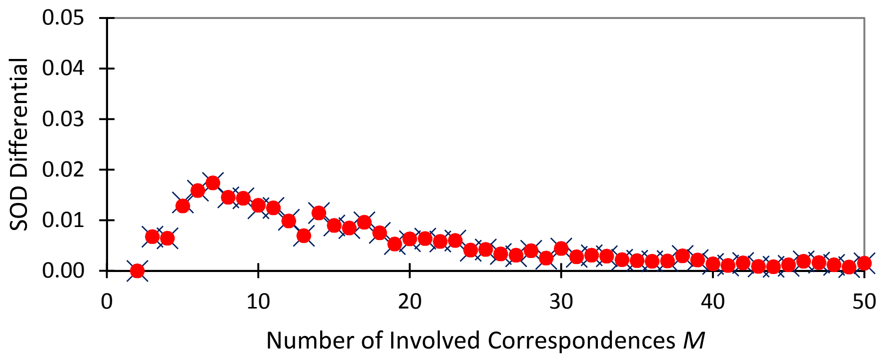
When this step is finished, the whole algorithm iterates until one of the following three options happens: 1) The sum of the minimum SOD of the whole correspondence in the set is lower than a threshold. 2) A maximum number of iterations I_{max} is achieved. 3) A minimum difference on the total SOD between the previous iteration and the current one is achieved. Independently of the terminating option, the algorithm returns the correspondence in the set that has at the moment the minimum SOD to the set as the GM correspondence. Convergence is assured since the SOD, in each iteration, is equal or lower than the previous iteration. Moreover, in case the SOD is kept equal, the algorithm stops.

3.4 Experimental Validation

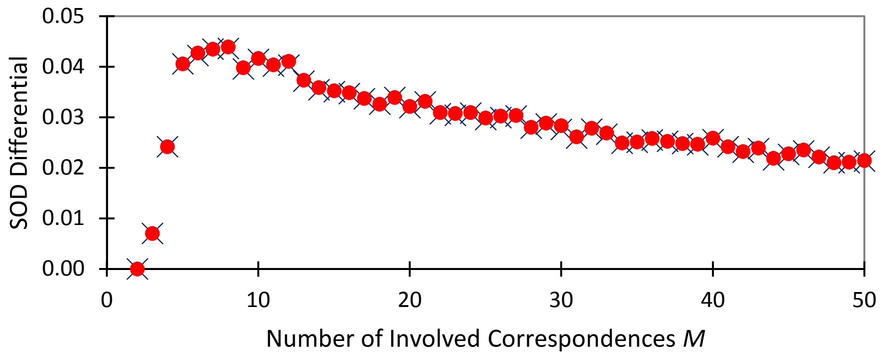
In this section, we show how close the suboptimal method is with respect to the optimal one, as well as the runtime of both algorithms. To have a fair comparison, we have used the HD in both cases.

We performed three tests, all of them executed the same way but using correspondences with $N = 5$, $N = 10$ and $N = 30$ mapped inlier elements in each case. Each test was prepared as follows. We randomly generated 100 sets of $M = 2,3 \dots ,50$ correspondences. For each set, both algorithms to find the GM correspondence are executed. In the minimisation method, the Hungarian method [4] was used to solve the LAP.

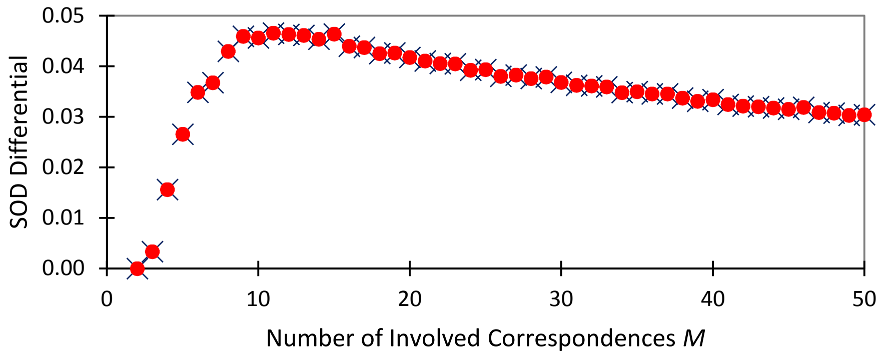
Figure 3.4 shows the normalised difference of SOD that the GM correspondences generated by the evolutionary method ($\Omega = 3$ for all cases) obtained with respect to the ones from the minimisation method in the first test ($N = 5$), second test ($N = 10$) and third test ($N = 30$) respectively. Each dot in the plot represents the average of the 100 executions. For the evolutionary method, we show results using $I_{max} = 1$ and $I_{max} = 2$. Results for larger values of I_{max} are not shown since they deliver exactly the same values that the ones of $I_{max} = 2$. In the three cases for $M = 2$, the evolutionary method obtains optimal GM correspondences, since the method only has to deal with the mean calculation. Nonetheless as M increases, this overestimation has a peak maximum value, and then it decreases until lowering down again towards the optimal value of the GM correspondence. This leads us to think that the evolutionary method has an optimal number of correspondences to be used, since certain values of M lead to more overestimation than others. Finally, from these plots we conclude that the evolutionary method, regardless of the I_{max} value used, obtains values that are really close to the optimal ones. In fact, the worst case overestimates the SOD in 4.5% with respect to the optimal SOD.



a) First test ($N = 5$)

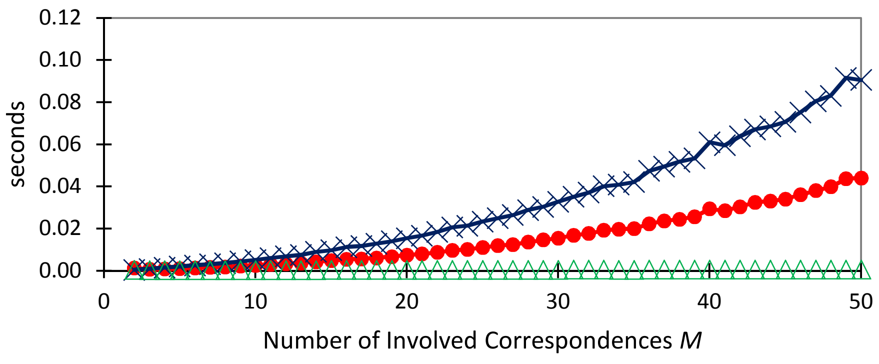


b) Second test ($N = 10$)

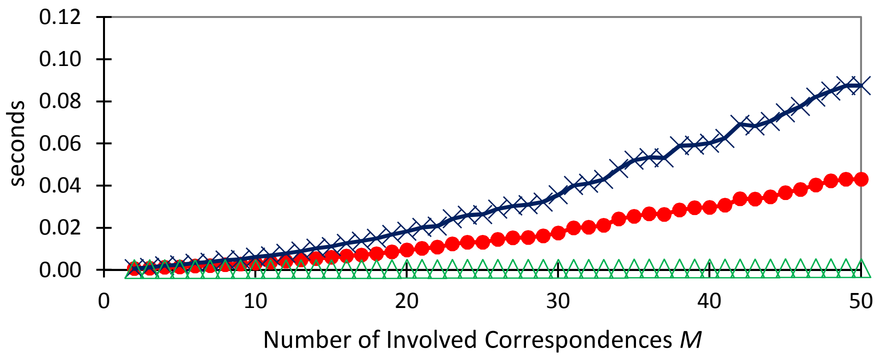


c) Third test ($N = 30$)

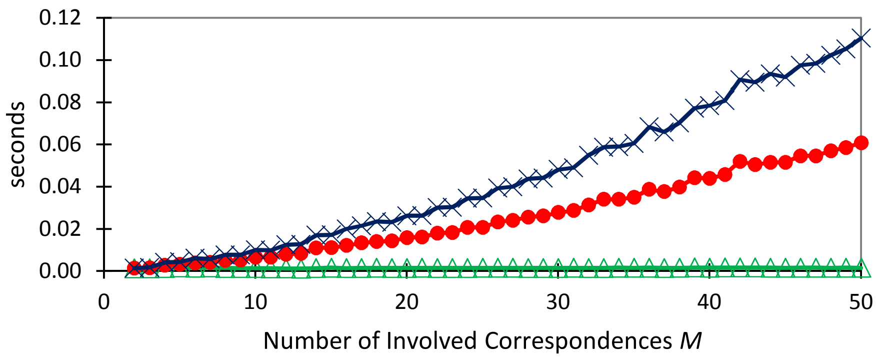
Figure 3. 4. Average difference of SOD between the evolutionary method and minimisation method (optimal algorithm) for a) $N = 5$, b) $N = 10$ and c) $N = 30$; (●) evolutionary method, $I_{max} = 1$, (×) evolutionary method, $I_{max} = 2$.



a) First test ($N = 5$)



b) Second test ($N = 10$)



c) Third test ($N = 30$)

Figure 3.5. Average runtime for a) $N = 5$, b) $N = 10$ and c) $N = 30$; (●) evolutionary method, $I_{max} = 1$, (×) evolutionary method, $I_{max} = 2$, (Δ) minimisation method.

Figure 3.5 shows the runtime spent by both approaches (in seconds). In the case of the evolutionary method, we show the two I_{max} figurations, where it is clear that the runtime increases as this parameter is set higher. When comparing both methods, the minimisation one is clearly faster than the evolutionary one, although both have a polynomial computational cost with respect to M. Finally, comparing the three plots between them, we realise the number of elements in the sets seems to have almost no influence on the runtime. Tests were performed using a PC with Intel 3.4 GHz CPU and Windows 7 as operative system.

3.5 Discussion

In this chapter, we have presented two methods to deduce the GM correspondence. For the first one, we have demonstrated that it obtains the exact median correspondence when all mapped elements of the output object are inliers and the HD is used. For the second one, it is shown that a fair approximation can be obtained. Nevertheless, the second method could be used with any type of elements and/or distance function between correspondences.

The evolutionary method has been presented as a viable solution for the approximation of the GM for data structures such as strings and graphs, since these cases imply second order relations and therefore, finding their exact GM has an exponential cost. However, In the concrete case that the aim is to find the exact GM of correspondences between inlier elements and the HD can be used, we have shown that this problem can be solved in cubic time (the computational cost of the LAP). Although this scenario seems rather limited, the importance of this chapter is to use this methodology as a first step towards the definition of a more robust method to learn a consensus correspondence. In the following chapters, a new proposal for a consensus framework will be presented, with the aim of learning the consensus correspondence in a more comprehensive form.

Still, it must be remarked that for the case of the GM calculation of correspondences, other distances between correspondence that take into consideration not only the element-to-element mapping, but also the structure and attributes of the related elements, could produce more interesting GM correspondences from the application point of view. For instance, in the case that correspondences relate attributed graphs, the mappings are defined node-to-node. Therefore, the distance function could consider the local structure of the nodes (its adjacent edges and their terminal nodes) to penalise the cost of a mapping. Then, a comparison between GM computation methods using the HD against methods using a distance of this nature would be interesting, not only in terms of runtime, but also in terms of accuracy. In chapter 8 of the present work, a new distance between correspondences called Correspondence Edit Distance will be defined to assess this problematic in the near future.

Chapter 4 Weighted Mean Consensus of a Pair of Correspondences

4.1 Introduction

In this chapter, the first step towards defining a new consensus framework for learning the consensus of two correspondences is introduced. This framework will be referred as the two-consensus framework (2-CF). It consists mainly of two key factors. On the one hand, it should model the consensus correspondence as a weighted mean of the two inputs, therefore the definition of the 2-CF is strongly inspired by the work previously done on finding the weighted mean of a pair of strings [1], graphs [2] and data clusters [3], as well as the definitions provided for the weighted mean of a pair of correspondence (chapter 2). Notice that the 2-CF must not restrict the consensus correspondence to be a strict mean but rather a weighted mean, given the flexibility of the latter concept. On the other hand, it should also be able to minimise the cost of the consensus correspondence to the most.

One of the main drawbacks of intending to learn a correspondence that accomplish both of these conditions resides in the large number of possible solutions. Furthermore, when the two input correspondences increase in size (number of mappings), the number of combinations which accomplish the weighted mean condition increases as well. For this purpose, the concept of optimisation becomes a viable solution. By using this approach, the combinatorial problem is converted into an assignment problem. Additionally, the cost-based optimisation given the features used in the extraction process can be considered simultaneously.

4.2 Definitions and Notations

Let $G = (\Sigma, \Upsilon)$ and $G' = (\Sigma', \Upsilon')$, of orders N and N' respectively, be sets of elements matched through correspondences f^1 and f^2, where the sets are composed of $m_i \in \Sigma$ (Σ denotes the domain of elements) and $\gamma_i \in \Upsilon$ (Υ denotes the domain of attributes of the elements). Similar definitions hold for G'. Based on the definition of the cost established in chapter 2, the cost between sets G and G' can be expressed through the use of a cost matrix $\boldsymbol{C}[i,j] = c(a_i, a'_j)$, where c is a cost function between the attributes of elements m_i and m'_j. In addition, suppose $c \in [0,1]$ to the definition of some required settings easier. Since most of the state of the art matching methods aim to produce an optimal correspondence f^* (the one that delivers the minimum cost), the linear minimisation problem of finding this minimum cost can be converted into the assignment problem [4] through the use of $\boldsymbol{C}$.

$$f^* = argmin_{\forall f}\ \{\boldsymbol{C}_f\} \tag{4.1}$$

where $\boldsymbol{C}_f$ is the cost of correspondence f obtained from matrix $\boldsymbol{C}$. That is

$$\boldsymbol{C}_f = \sum_{i=1}^{N}\sum_{j=1}^{N\prime} \boldsymbol{C}[i,j]\ \forall f(a_i) = a'_j$$
$$s.t. \forall\, i\ \neq k, f(a_i) \neq f(a_k) \tag{4.2}$$

Secondly, it is also desired to learn a consensus which is close to the mean correspondence $\bar{f}$. Given that the cost minimisation is not related to the mean, then a weighted mean, or at least a good approximation to be a weighted mean, is a desirable property for the consensus correspondence. To achieve this, it is possible to convert the correspondences to structures similar to the correspondence matrix F presented in chapter 3. The fact that this matrix can be used to find a not only the GM, but also the weighted mean, will be demonstrated later in this chapter.

As a result of both requisites, a standard minimisation approach (SMA) must be established, with the aim of learning an optimal correspondence $\bar{f}^*$ that globally minimises the two functions. The SMA that adapts more to the proposed scenario is composed of a loss function $\nabla(e)$ plus a regularization term $\Omega(e)$ [5], weighted by a parameter λ [6]. The loss function is the cost function to be minimised, and the regularisation term is the mathematical mechanism that imposes the restriction, namely the weighted mean condition. Parameter λ weights how much these terms have to be considered.

$$e^* \ = argmin_{\forall e}\ \{\nabla(e) +\ \lambda \cdot \Omega(e)\} \tag{4.3}$$

The 2-CF intends to learn $\bar{f}^*$ such that the following equation holds,

$$\bar{f}^* \ = argmin_{\forall f}\ \{\ \lambda_{\boldsymbol{C}} \cdot \nabla(f) +\ \lambda_F \cdot \Omega(f)\} \tag{4.4}$$

Although not strictly necessary, equation 4.4 is presented with parameters λ_C and λ_H instead of only one parameter λ as in equation 4.3 to simplify the presentation of the results in the experimental section.

To facilitate the explanation of the 2-CF, the example provided in figure 4.1 will be commented throughout this chapter. Suppose two sets G and G' of 7 elements have attributes represented by natural numbers (table 4.1). Additionally, two correspondences f^1 and f^2 have been defined between these two sets.

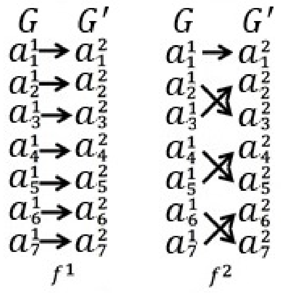


Figure 4. 1. Correspondences f^1 and f^2 between sets G and G'.

Table 4. 1. Attributes of sets G and G'.

Sets	γ_1, γ_1'	γ_2, γ_2'	γ_3, γ_3'	γ_4, γ_4'	γ_5, γ_5'	γ_6, γ_6'	γ_7, γ_7'
G	1	4	7	5	3	2	2
G'	2	5	6	8	4	3	2

The HD between f^1 and f^2 is $d_H(f^1, f^2) = 6$, with only the first mapping being common between both correspondences, thus $f^1(a_1) = f^2(a_1)$. Assume a cost function between elements a_i^1 and a_j^2 of both sets defined as $c(\gamma_i, \gamma_j') = |\gamma_i - \gamma_j'|/max(\gamma_i, \gamma_j')$. Therefore, the total costs of the correspondences are $Cost(G, G', f^1) = 1.8$ and $Cost(G, G', f^2) = 2.3$.

4.3 Methodology

As mentioned in the beginning of this chapter, the proposed framework learns the optimal correspondence $\bar{f}^*$ through the use of a SMA function like the one in equation 4.4, in which the loss function and the regularisation term can be represented as

$$\nabla(f) = Cost(G, G', f) \text{ and } \Omega(f) = d_H(f^1, f) + d_H(f, f^2) - d_H(f^1, f^2) \tag{4.5}$$

Since the aim is to learn a correspondence from exclusively two initial correspondences, the framework will only focus on seeking for a partial correspondence where both of the involved parties disagree. The other partial correspondence, in which both decided the same point mapping, is directly assigned to the final result. Therefore, both correspondences are split into two disjoint sub-correspondences such that $f^1 = f'^1 \cup f''^1$ and $f^2 = f'^2 \cup f''^2$. Sub-correspondences f'^1 and f'^2 are related with the mappings where $f^1(a_i) = f^2(a_i)$, and f''^1 and f''^2 are related with the ones where $f^1(a_i) \neq f^2(a_i)$. This also implies that the cost of both correspondences is $Cost(G,G',f^1) = Cost\big(G,G',f'^1\big) + Cost(G,G',f''^1)$ and $Cost(G,G',f^2) = Cost\big(G,G',f'^2\big) + Cost(G,G',f''^2)$. Additionally, $\Sigma = (\Sigma)' \cup (\Sigma)''$. The set of elements $(\Sigma)'$ in G is composed of the elements such that $f^1(a_i) = f^2(a_i)$, while the set of elements $(\Sigma)''$ in G is composed of the elements such that $f^1(a_i) \neq f^2(a_i)$. As a result, the obtained consensus correspondence $\bar{f}^*$ will be a union of two sub-correspondences, $\bar{f}^* = \bar{f}'^* \cup \bar{f}''^*$, where $\bar{f}'^* = f'^1 = f'^2$ and $\bar{f}''^*$ is defined by the following equation:

$$\bar{f}''^*_{\lambda_C,\lambda_F} = argmin_{\forall f''}\left\{\begin{array}{c} \lambda_{\boldsymbol{C}} \cdot Cost(G,G',f'') + \\ \lambda_F \cdot \big[\, d_H\big(f''^1,f''\big) + d_H\big(f'',f''^2\big) - d_H\big(f''^1,f''^2\big)\big] \end{array}\right\} \tag{4.6}$$

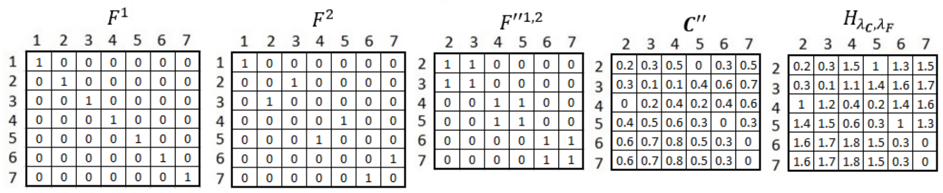
If $\lambda_{\boldsymbol{C}} = 0$ and $\lambda_F > 0$, then the consensus correspondence will be the median of both correspondences, as shown in the previous chapter. Conversely, if $\lambda_C > 0$ and $\lambda_F = 0$, then the method tends to minimize the cost without taking into account the initial proposals. To solve equation 4.6, a linear assignment problem (LAP) such as [7], [8] and [9] is used to convert the minimisation problem into an assignment problem, which consequentially minimises expression H_{λ_C,λ_H} defined as

$$H_{\lambda_C,\lambda_H} = \lambda_{\boldsymbol{C}} \cdot \boldsymbol{C}''_{f''} + \lambda_F \cdot [\mathbf{1} - F''^{1,2}]_{f''} \tag{4.7}$$

where $[\cdot]_{f''}$ denotes the cost of the correspondence f'' applied on matrix $[\cdot]$ (equation 4.2). Moreover, $\boldsymbol{C}''[i,j] = c(a_i, a'_j)$ with $a_i \in (\Sigma)''$ and $a'_j \in (\Sigma')''$. Besides, $F''^{1,2} = F''^1 + F''^2$, where F''^1 and F''^2 are the correspondence matrices corresponding to f''^1 and f''^2 respectively. In addition, $\mathbf{1}$ represents a matrix of all ones. Note that the number of rows and columns of matrices $\boldsymbol{C}''$, F''^{a} and F''^{b} must be lower or equal than $\boldsymbol{C}$. The more similar the node mappings of f^1 and f^2 are, the smaller the number nodes in $(\Sigma)''$ and $(\Sigma')''$ is, and so, the dimensions of $\boldsymbol{C}''$, F''^{a} and F''^{b}. This fact affects directly on the computational cost of the method. Considering equation 4.7, we obtain the following expression:

$$\bar{f}''^{*}_{\lambda_C,\lambda_H} = argmin_{\forall f''}\left\{\left(H_{\lambda_C,\lambda_F}\right)_{f''}\right\} \tag{4.8}$$

Figure 4.2 shows values of matrices F^1, F^2, $F''^{1,2}$, $\boldsymbol{C}''$ and H_{λ_C,λ_F} (using $\lambda_C = 1$ and $\lambda_F = 1$ on the practical example). Notice that after excluding the first mapping, the size of $F''^{1,2}$, $\boldsymbol{C}''$ and H_{λ_C,λ_F} is reduced.



F^1

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	0	1	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0
7	0	0	0	0	0	0	1

F^2

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0
3	0	1	0	0	0	0	0
4	0	0	0	0	1	0	0
5	0	0	0	1	0	0	0
6	0	0	0	0	0	0	1
7	0	0	0	0	0	1	0

$F''^{1,2}$

	2	3	4	5	6	7
2	1	1	0	0	0	0
3	1	1	0	0	0	0
4	0	0	1	1	0	0
5	0	0	1	1	0	0
6	0	0	0	0	1	1
7	0	0	0	0	1	1

$\boldsymbol{C}''$

	2	3	4	5	6	7
2	0.2	0.3	0.5	0	0.3	0.5
3	0.3	0.1	0.1	0.4	0.6	0.7
4	0	0.2	0.4	0.2	0.4	0.6
5	0.4	0.5	0.6	0.3	0	0.3
6	0.6	0.7	0.8	0.5	0.3	0
7	0.6	0.7	0.8	0.5	0.3	0

H_{λ_C,λ_F}

	2	3	4	5	6	7
2	0.2	0.3	1.5	1	1.3	1.5
3	0.3	0.1	1.1	1.4	1.6	1.7
4	1	1.2	0.4	0.2	1.4	1.6
5	1.4	1.5	0.6	0.3	1	1.3
6	1.6	1.7	1.8	1.5	0.3	0
7	1.6	1.7	1.8	1.5	0.3	0

Figure 4. 2. Correspondence matrices F^1 and F^2 from correspondences f^1 and f^2 and matrices $F''^{1,2}$, $\boldsymbol{C}''$ and H_{λ_C,λ_H} used by the framework (using $\lambda_{\boldsymbol{C}} = 1$ and $\lambda_F = 1$).

In the next section, it is demonstrated that equation 4.8 minimise the same approximation than equation 4.6 for all weights $\lambda_{\boldsymbol{C}}$ and λ_F and pairs of graphs G and G'. Additionally, the cases in which the learnt consensus correspondence is an exact weighted mean or an approximate weighted mean are specified.

4.4 Reasoning about Optimality

To say that equation 4.8 solves the problem that equation 4.6 formulates, it must be demonstrated that function $\{\lambda_{\boldsymbol{C}} \cdot Cost(G, G', f'') + \lambda_F \cdot [d_H(f''^1, f'') + d_H(f'', f''^2) - d_H(f''^1, f''^2)]\}$ extracted from equation 4.6 minimises the same partial correspondence than $\left\{\left[\lambda_C \cdot \boldsymbol{C}''_{f''} + \lambda_F \cdot [1 - F''^{1,2}]\right]_{f''}\right\}$ extracted from equation 4.7. First, notice that by definition of the cost (chapter 2), $Cost(G, G', f'') = \boldsymbol{C}''_{f''}$. Therefore, only the following equality must be demonstrated:

$$[1 - F''^{a,b}]_{f''} = d_H(f''^1, f'') + d_H(f'', f''^2) - d_H(f''^1, f''^2) \qquad (4.9)$$

Suppose the cardinality of $(\Sigma)''$ and $(\Sigma')''$ is n. Since these sets do not have any element in common, then by definition, $d_H(f''^1, f''^2) = n$. Given the involved correspondences f'', f''^1 and f''^2, three natural numbers n_p, n_q and n_t can be defined:

1) n_p: number of nodes in $(\Sigma)''$ that hold $f''(a_i) \neq f''^1(a_i)$ and $f''(a_i) \neq f''^2(a_i)$.

2) n_q: number of nodes in $(\Sigma)''$ that hold $f''(a_i) = f''^1(a_i)$ and $f''(a_i) \neq f''^2(a_i)$.

3) n_t: number of nodes in $(\Sigma)''$ that hold $f''(a_i) \neq f''^1(a_i)$ and $f''(a_i) = f''^2(a_i)$.

Likewise, by definition of these sets, there is no m_i such that $f''(a_i) = f''^1(a_i)$ and $f''(a_i) = f''^2(a_i)$. Therefore, $n = n_p + n_q + n_t$. For simplicity, the nodes in $(\Sigma)''$ are ordered such that a_1 to a_{n_p} hold the first condition, a_{n_p+1} to $a_{n_p+n_q}$ hold the second condition, and $a_{n_p+n_q+1}$ to a_n hold the third condition. For equation 4.9 to be valid, $[1 - F''^{1,2}]_{f''} = n_p$ must hold, and likewise $d_H(f''^1, f'') + d_H(f'', f''^2) - d_H(f''^1, f''^2) = n_p$.

1) Demonstration $[1 - F''^{1,2}]_{f''} = n_p$:

Suppose that $f''(m_i) = m'_j$, then

$$[1 - F''^{1,2}]_{f''} = \sum_{i=1}^{n} (1 - F''^{1,2})[i,j] = \sum_{i=1}^{n_p} 1 + \sum_{i=n_p+1}^{n} 0 = n_p \qquad (4.10)$$

2) Demonstration $d_H(f''^1, f'') + d_H(f'', f''^2) - d_H(f''^1, f''^2) = n_p$:

$$\begin{aligned} d_H(f''^1, f'') + d_H(f'', f''^2) - n &= \sum_{i=1}^{n}\Big(2 - \delta\left(f''^1(a_i), f''(a_i)\right) \\ &\quad - \delta\left(f''(a_i), f''^2(a_i)\right)\Big) - n \\ &= \left(\sum_{i=1}^{n_p} 2 + \sum_{i=n_p+1}^{n_p+n_q} 1 + \sum_{i=n_p+n_q+1}^{n} 1\right) - n \\ &= 2n_p + n_q + n_t - n = n_p \end{aligned} \tag{4.11}$$

■

In some cases, it is also interesting to know if the learnt consensus correspondence is either an exact weighted mean or an approximated one. First, consider that if both f' and f'' are partial weighted mean correspondences, then the union $f = f' \cup f''$ would also be a weighted mean correspondence, since by definition $f' \cap f'' = 0$. It is clear that if the weighted mean condition holds for both partial correspondences, then it holds for the complete one. Moreover, f' is always defined as weighted mean correspondence. Therefore, it is stated that the learnt correspondence $\bar{f}^*$ is a weighted mean correspondence when $\bar{f}''^*$ is also a weighted mean correspondence. These cases are the ones where $\bar{f}''^*$ holds the weighted mean condition. Since it has been demonstrated that $d_H(f''^1, f'') + d_H(f'', f''^2) - d_H(f''^1, f''^2) = n_p$, then n_p must be 0. By definition of n_p, these correspondences are the ones where $\bar{f}''^*(a_i) = f''^1(a_i)$ or $\bar{f}''^*(a_i) = f''^2(a_i)$. Therefore, the conclusion is that:

$$\bar{f}^*_{\lambda_C,\lambda_F} \text{ is a weighted mean correspondence if}$$

$$\bar{f}''^*_{\lambda_C,\lambda_F}(a_i) = \begin{cases} f''^1(a_i) & or \\ f''^2(a_i) & \forall\, a_i \in (\Sigma)'' \end{cases} \tag{4.12}$$

The cost of testing if the correspondence obtained is a weighted mean is linear with respect to the number of discrepancies between correspondences f^1 and f^2.

Note that if $\lambda_C = 0$ and $\lambda_F > 0$ and a LAP solver [7], [8] or [9] is used to solve equation 4.8, the framework always obtains a weighted mean correspondence. That is because the solvers guarantee to assign either of the two mappings on the final consensus correspondence without inserting any additional mappings. As discussed in chapter 2, the condition required to produce weighted mean correspondences is $\bar{f}_w(a_x) = f^1(a_x)$ or $\bar{f}_w(a_x) = f^2(a_x)$ for all mappings of non-null elements a_x of the output set of elements.

4.5 Experimental Validation

Once the databases used are properly described, a first experiment which aims to confirm the theoretical background and to validate parameters λ_C and λ_F is presented using an artificially created joint set scenario. This experiment will allow us to measure the learnt consensus correspondence's cost and the proximity to be a weighted mean. Afterwards, to demonstrate a first insight on how the 2-CF could cope with the disjoint set scenario, two experiments are presented. The first one uses inlier mappings only, while the second one uses every possible mapping.

4.5.1 Databases Used

-*Tarragona Palmprint*: This database consists of triplets composed of two sets of minutiae (salient points of a palmprint image) G_i and G_i', and a correspondence f_i between them. The database was created as follows. First, we collected images contained in the Tsinghua 500 PPI Palmprint Database [10], a public high-resolution palmprint database composed of the palmprints of 300 subjects, with a 2040×2040 pixel resolution. The original dataset provides a total of 8 different palmprint inks (pictures) per subject. We used only the first 10 subjects of the database, therefore 80 palmprint images compose our dataset. For each image, the respective minutiae set were extracted using the algorithm presented in [11]. The average number of extracted minutiae is 836.3 minutiae per palmprint. Given each subject, the minutiae correspondences between the 8 sets of minutiae are computed through a greedy matching algorithm based on the Hough Transform [12]. Thus, we generate a total of 64 ground truths per subject and so, a total of 640 triplets composed of two sets of minutiae from the same subject and their respective correspondences, $\{G_i, G_i', f_i\}$, where $i \in [1..640]$. The information of each minutia is the 2D position, angle, type of minutia (terminal or bifurcation) and quality (good or poor). Figure 4.3 shows a sample of a palmprint and its minutiae extracted.

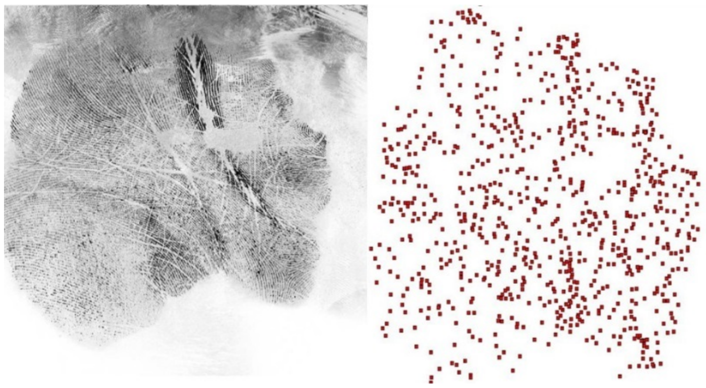


Figure 4.3. A palmprint image from the "Tarragona Palmprint" database (left) and the minutiae extracted from it (right).

- *Tarragona Exteriors*: This database is composed of 5 sequences × 10 pairs of images × 5 salient point and feature extractors × 2 matching algorithms = 500 quartets of two sets of salient points and two correspondences between them, which are split among 10 datasets (5 extractors × 2 matching algorithms). The first correspondence has been obtained with an existing matching method, and the second one is the ground truth correspondence. To construct this database, we used the 11 image samples from 5 public image databases called "BOAT", "EASTPARK", "EASTSOUTH", "RESIDENCE" and "ENSIMAG" [13]. The original databases are composed of sequences of images taken of the same object, but from different camera angles and positions. Together with the images, the homography estimations that convert the first image (img00) of the set into the other ones (img01 through img10) is provided. From each of the images, we extracted the 50 most reliable salient points using 5 methodologies: FAST [14], HARRIS [15], MINEIGEN [16], SURF [17] (native Matlab 2013b libraries) and SIFT [18] (own library). A total of 10 correspondences between the first image of the sequence and the other ten ones is computed through the use of two algorithms: Matlab's *matchFeatures* function [19] (with the parameter $MaxRatio = 1$ to ensure the maximum number of inlier mappings) and the FBP method [20] (with an insertion and deletion cost of $K_v = 0.2$ that allowed a fair number of inlier mappings). Finally, we construct the ground truth correspondence using the homography provided by the original image databases. In summary, the database has a total 10 datasets with a total of 500 quartets (50 quartets per dataset) composed of two sets of features and two correspondences $\{G_i, G'_i, f_i, h_i\}$, where $i \in [1..500]$. Figure 4.4. shows a sample of the first image in each sequence, along with some SIFT salient points over the image.

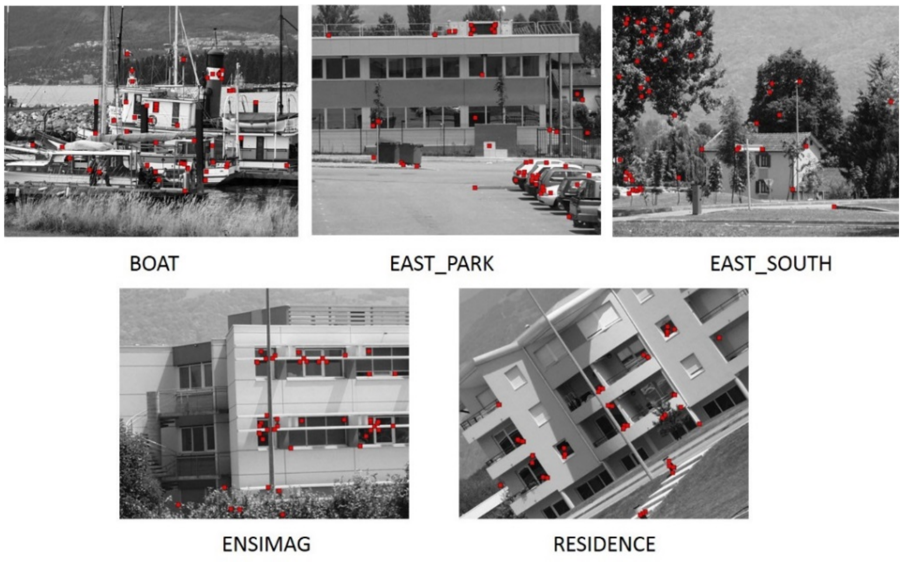


Figure 4.4. The first image of each sequence of the "Tarragona Exteriors" database and the SIFT salient points extracted from them.

Both databases are available in [21].

4.5.2 Theoretical Validation in a Joint Set Scenario

The following test is performed in the first database, given the previously tested positive results [10], [11], [12] of the proposed ground truth correspondences. Two different aspects of the 2-CF will be evaluated. First, we analyse how minimum the cost of the obtained consensus is with respect to the ground truth correspondence. To that aim, a cost function which considers the distance between the angles and the positions of the minutiae is used (this cost function has been described in chapter 2). By definition, the costs of the learnt consensus correspondence $\bar{f}^*_{\lambda_C,\lambda_{F_i}}$ and the ground truth correspondence f_i can be calculated as the sum of the costs of all their mappings and can be compared by subtracting them. Second, we analysed if the proposed framework is minimising the HD and approaching the result to be a weighted mean. For this purpose, a regularisation metric is used as in equation 4.13. As the value of $Regularisation$ approaches to 1, the closer a consensus is to be a weighted mean.

$$Regularisation\left(f^1, f^2, \bar{f}^*_{\lambda_C,\lambda_F}\right) = \frac{d_H\left(f^1, \bar{f}^*_{\lambda_C,\lambda_F}\right) + d_H\left(f^2, \bar{f}^*_{\lambda_C,\lambda_F}\right)}{d_H\left(f^1, f^2\right)} \qquad (4.13)$$

Given each ground truth correspondence f_i in the 640 triplets that compose the "Tarragona Palmprint" database, 212 pairs of correspondences f_i^1 and f_i^2 are generated by randomly adding noise to f_i. Therefore, a total of $640 \times 212 = 135'680$ pairs of correspondences: $\{f_i^{\alpha 1}, f_i^{\alpha 2}\}$ have been tested, where $\mathrm{i} \in [1, \dots, 640]$ and $\alpha \in [1, \dots, 212]$. Index i represents the original triplet, and α represents the level of noise added to the original correspondence in the i^{th} triplet. The noise is set such that $d_H\left(f_i^{\alpha 1}, f_i^{\alpha 2}\right) = \alpha$. Thus, $f_i^{\alpha 1}$ is first generated in a random form, and then $f_i^{\alpha 2}$ is constructed such that $d_H\left(f_i^{\alpha 2}, f_i\right) = (\alpha - \beta)$, where $0 \leq \beta \leq \alpha$ and $\beta = d_H\left(f_i^{\alpha a 1}, f_i\right)$. This is to avoid $d_H\left(f_i^{\alpha 1}, f_i\right) = d_H\left(f_i, f_i^{\alpha 2}\right)$ for every case.

Figures 4.5 to 4.6 show the obtained results for this validation. The tree most illustrative configurations for $\lambda_{\boldsymbol{C}}$ and λ_F have been tested: a) $\lambda_{\boldsymbol{C}} = 1\,; \lambda_F = 0$. This configuration reproduces a classical minimum-cost method (red lines), since the involved correspondences are not considered at all. b) $\lambda_{\boldsymbol{C}} = 1$; $\lambda_F = 1$. There is an equal contribution of both the cost and the involved correspondences, keeping in mind that costs are normalised (green lines). c) $\lambda_{\boldsymbol{C}} = 0$; $\lambda_F = 1$. The cost is not considered and therefore, this approach would be similar to the GM calculation shown in chapter 3 (blue lines). The horizontal axis in these figures represents the noise α, while the vertical axis represents the average difference in cost for the 640 triplets.

Figure 4.5 shows the average cost difference between $\bar{f}^*_{\lambda_C,\lambda_{F_i}}$ and f_i as the noise is increased for the three proposed weight configurations. It is clearly noticeable that, the larger the distance between the correspondences $\alpha = d_H\left(f_i^{\alpha 1}, f_i^{\alpha 2}\right)$, the higher the cost of the resulting consensus correspondence. This relation appears because when the involved correspondences were generated from the original ground truth in the "Tarragona Palmprint" database, some random modifications that made the correspondence "worse" than the original one are introduced, which translates in a cost increase. Also, it is clear that the cost of $\bar{f}^*_{\lambda_C,\lambda_{F_i}}$ is lower when the optimisation method considers the cost variable $\lambda_{\boldsymbol{C}} = 1$ (red and green lines) and so, we deduce that this term must be always considered in the optimisation function. Moreover, the cost of configuration $\lambda_{\boldsymbol{C}} = 1\,; \lambda_F = 1$ (green line) is slightly higher than configuration $\lambda_{\boldsymbol{C}} = 1\,; \lambda_F = 0$ (red line). This happens because intending to force the consensus correspondence to also be a weighted mean increases slightly the correspondence's cost.

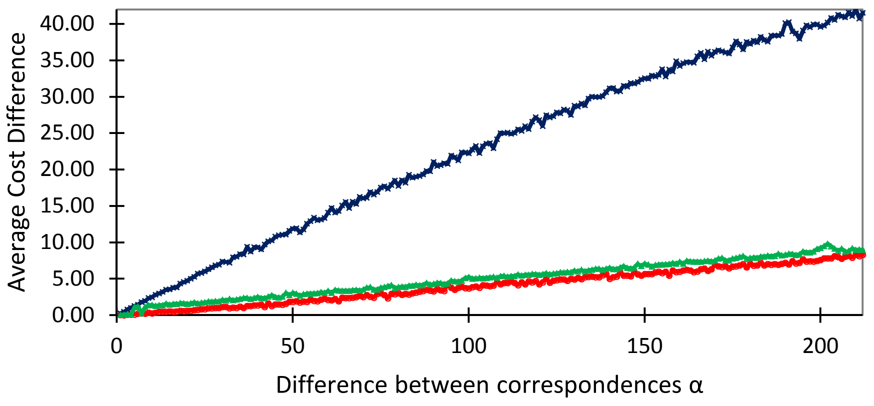


Figure 4. 5. Cost difference between $\bar{f}^{*}_{\lambda_C,\lambda_{F_i}}$ and f_i as α is increased; (🟥) $\lambda_C = 1\,;\, \lambda_F = 0$, (🟦) $\lambda_C = 0\,;\, \lambda_F = 1$, (🟩) $\lambda_C = 1\,;\, \lambda_F = 1$.

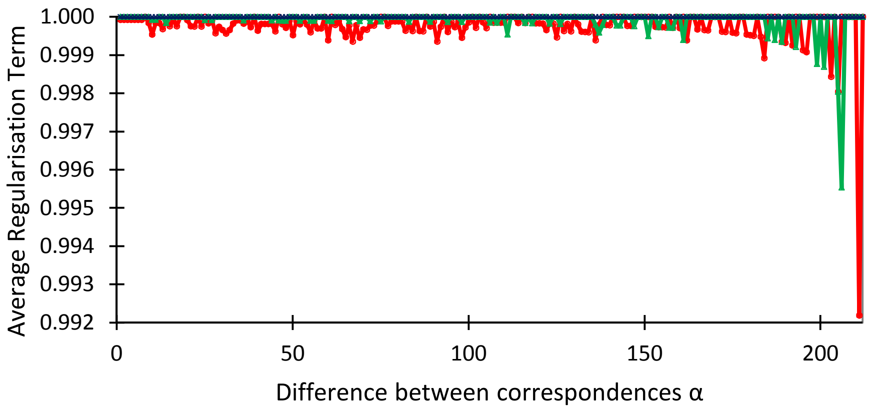


Figure 4. 6. Regularisation term minimisation for $\bar{f}^{*}_{\lambda_C,\lambda_{F_i}}$ as α is increased; (🟥) $\lambda_C = 1\,;\, \lambda_F = 0$, (🟦) $\lambda_C = 0\,;\, \lambda_F = 1$, (🟩) $\lambda_C = 1\,;\, \lambda_F = 1$.

Figure 4.6 shows the behaviour of the regularisation term (equation 4.13) as the differential α is increased. As stated before, the $\lambda_C = 0\,;\, \lambda_F = 1$ configuration always selected a weighted mean consensus, and thus, the value of $Regularisation$ remains 1 throughout the whole experiment. As for the other two cases, both configurations are slightly less likely to obtain a weighted mean correspondence, but in general, the they always obtain an average value which never decreases below $Regularisation = 0.99$. This means that regardless the configuration, when the 2-CF is implemented with any weight configuration, there is a high likelihood to obtain weighted mean consensus correspondences.

From these first experiment, it is demonstrated that the proposed framework is valid, and that when both parameters are taken into consideration, the obtained consensus correspondences are more likely to be an approximate, weighted, or even an exact mean, and their costs are closer to the minima.

4.5.3 Validation with Inlier Mappings in a Disjoint Set Scenario

The experiments in this subsection were performed using the "Tarragona Exteriors" database, using only the datasets that contained correspondences generated using the *matchFeatures* function (250 quartets). The 2-CF was applied using the configuration $\lambda_{\boldsymbol{C}} = 1$; $\lambda_F = 1$, and the aim was to validate the number of correct element mappings comparing the obtained result with the ground truth correspondence. The cost function used was the Euclidean distance between the two sets of features. Since all the feature vectors in the database are normalised and have the same size, in this first approach it is assumed that it is fair to compare features from different extractors. We compared the 2-CF with a simpler framework called No Discrepancies (ND), which conforms the final correspondence using the mappings that the two initial correspondences agree on.

Figure 4.7 presents a graphical representation of the correct inlier mappings (deduced by comparing with the ground truth correspondences) of two correspondences between the first two images in the "Tarragona Exteriors" database (BOAT sequence). A first correspondence was obtained using the FAST feature extractor and the *matchFeatures* matching algorithm, while the second one is generated using the HARRIS feature extractor and the same matching algorithm. In red we show the salient points and correspondences made exclusively by the first correspondence, and in yellow we show the salient points extracted and the correspondences made by the second one. In addition, the salient points and mappings that appeared in both methods are depicted in green. Notice that in order to consider two salient points from different extractors as the same, a Euclidean distance smaller than 6 pixels must be obtained between any two given points. We realise that FAST has achieved 6 correct correspondences (3 red lines plus 3 green ones) and HARRIS has achieved 7 correct correspondences (4 yellow lines plus 3 green ones). In this case, the ND method would obtain 10 correct correspondences (the addition of red, green and yellow) since there are no differences between the red and yellow lines.

Points that appear unmapped in the figure are outliers, but for the case at hand will not be considered.

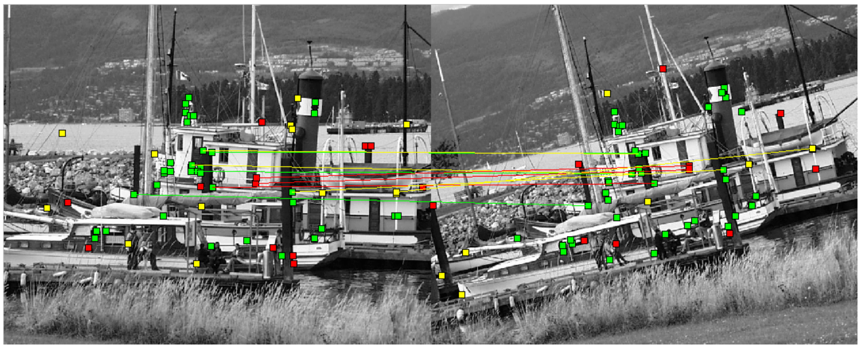


Figure 4.7. Two pairs of salient points and correspondences (red and yellow) between a pair of images, and their coincidences (green).

Correspondences f''^1_i and $f_i''^2$ are composed of red and yellow lines, and correspondences f'^1_i and $f_i'^2$ are represented by the green lines. Note that in each experiment, the cardinality of the sets will not depend on the original sets, but rather on the matching algorithms. Moreover, given the correspondences f''^1_i and $f_i''^2$, there are two types of points in both sets. The ones that contribute in both correspondences and the ones that only appear in one of them. Therefore, such points are outlier elements and thus, not considered by the correspondence that is not matching them.

Figure 4.8 shows the consensus correspondence obtained by the two-input consensus framework. Only the correct mappings with respect to the ground truth correspondence are depicted. For this pair of items, we also have achieved 10 correct point mappings.

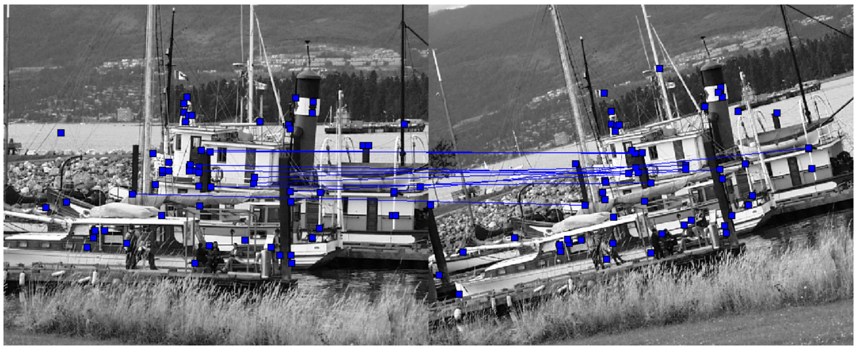


Figure 4. 8. Consensus correspondence obtained by the two-input consensus framework.

Table 4.2 shows the average number of correct correspondences obtained by the 2-CF compared to the ND method given the 5 sequences. $E_{a,b}$ represents the combination of feature extractors a and b, where 1: FAST, 2: HARRIS, 3: MINEIGEN, 4: SURF and 5: SIFT. Notice that the consensus framework obtains in average a greater number of correct correspondences with respect to the ND framework for all combinations and datasets.

Table 4. 2. Average number of correct inlier mappings obtained by the 2-CF and the ND method.

Dataset		$E_{1,2}$	$E_{1,3}$	$E_{1,4}$	$E_{1,5}$	$E_{2,3}$	$E_{2,4}$	$E_{2,5}$	$E_{3,4}$	$E_{3,5}$	$E_{4,5}$	Av.
1	ND	12	14	65	46	13	64	45	67	49	79	45.4
	2-CF	**14**	**16**	**70**	**53**	**14**	**69**	**54**	**73**	**59**	**95**	**51.7**
2	ND	3	3	67	74	1	68	61	70	63	213	62.3
	2-CF	**4**	**4**	**150**	**79**	**2**	**147**	**76**	**144**	**91**	**270**	**96.7**
3	ND	1	1	43	3	1	43	2	43	2	43	18.2
	2-CF	**1**	**1**	**48**	**6**	**1**	**48**	**5**	**48**	**5**	**50**	**21.3**
4	ND	1	1	68	4	2	63	6	65	4	73	28.7
	2-CF	**2**	**3**	**106**	**4**	**3**	**108**	**5**	**104**	**4**	**110**	**44.9**
5	ND	1	2	75	34	1	73	39	73	67	113	47.8
	2-CF	**1**	**2**	**112**	**54**	**1**	**111**	**53**	**111**	**75**	**153**	**67.3**
Average	ND	3.6	4.2	63.6	32.2	3.6	62.2	30.6	63.6	37	104	3.6
	2-CF	**4.4**	**5.2**	**97.2**	**39.2**	**4.2**	**96.6**	**38.6**	**96**	**46.8**	**135**	**4.4**

4.5.4 Validation with All Mappings in a Disjoint Set Scenario

For this experiment, "Tarragona Exterior" database was once again used, selecting the two most frequently used feature extractors in the consulted literature: SIFT and SURF. Moreover, all datasets which contained salient points from these two feature extractors were considered (correspondences generated with either matching algorithm). The 50 mappings of each correspondence are considered this time, and two correspondences are converted into mutually bijective ones before they are inserted into the 2-CF.

Table 4.3 shows the average number of differences between the two correspondences (generated by the *matchFeatures* matching algorithm and the FBP matching algorithm respectively), as well as the percentage that these discrepancies represent (considering the number of points in the set is 50). This table is shown since by definition; the framework can only increase the accuracy when the number of discrepancies is considerable. If there are no discrepancies, the algorithm imposes the mapping decided by both correspondences.

Table 4.3. Average number and percentage of discrepancies between the two correspondences.

Sequence	SIFT	SURF	SIFT (%)	SURF (%)
BOAT	17.1	39.7	34.2	79.4
EASTPARK	16	44.8	32	89.6
EASTSOUTH	17	40.8	34	81.6
RESIDENCE	19.7	44.3	39.4	88.6
ENSIMAG	18.5	43.3	37	86.6
AVERAGE	**17.6**	**42.5**	**35.3**	**85.1**

Table 4.4 shows the average accuracy of 2-CF for this situation. This value is computed through the accuracy of the 10 possible correspondences (img00-img01 to img00-img10) in each of the 5 sequences. The overall accuracy of a correspondence is the number of correct mappings divided by 50 possible ones (Absolute Accuracy). When the correspondences are considered in the optimisation function ($\lambda_F = 1$), 2-CF obtains slightly better accuracy than both of the initial matching algorithms. Subsequently, this accuracy is increased when the cost is considered as well ($\lambda_C = 1$). The accuracy on the discrepancies column is added to measure the fact that the framework can only increase the accuracy in the case of discrepancies (Discrepancies Accuracy column). This value represents the percentage of mappings that are properly found given that a discrepancy between the original methods existed. Notice that the proposed framework obtains an important increase in the absolute accuracy with respect to the initial methods. Moreover, an accuracy of 85% (SURF) and 47% (SIFT) is obtained for the Discrepancies Accuracy with respect to the original methods.

Table 4.4. Absolute Accuracy and Discrepancies Accuracy of each combination.

Feature Extractor	Matching Algorithm	Absolute Accuracy (%)	Discrepancies Accuracy (%)
SURF	*matchFeatures*	29.56	
	FBP	24.32	
	Consensus	30.28	85.73
SIFT	*matchFeatures*	39.56	
	FBP	25.6	
	Consensus	40.12	47.11

4.6 Discussion

In this chapter, a first step towards defining a consensus framework has been presented for the case where the number of involved correspondences is exactly two. The framework is based on a classical optimisation scheme as in the previous chapter, but this time, the cost yielded by the two involved correspondences is taken into consideration. So far, the 2-CF has been proven to be a more suitable solution when the consensus has to consider multiple factors, and not just the fact of being an arithmetical median of the initial proposals. Such is the case mostly in correspondences derived from image matching, where the these are generated through a cost minimisation process that is not related to the mean concept.

Three main issues arise from this initial approach. First, the disjoint set scenario has to be addressed properly. Second, given that many of the mappings are outlier ones, 2-CF only increases discrepancy accuracy, but not the absolute accuracy. This is strongly related with spurious mapping rejection; an issue solved through the use of optimisation approaches such as the Bipartite (BP) framework [22]. Finally, when the structural information of the mapped elements must be considered (i.e. graph correspondences), the consensus learning must be modelled so that the cost matrix also considers this information. For instance, information such as the node centralities used in the graph matching process can be added. Both aspects will be addressed in the following chapter.

Chapter 5
Consensus of a Pair of Correspondences through the Bipartite Framework

5.1 Introduction

From the work presented in chapter 4, three main issues have been discovered when intending to learn the consensus correspondence given two initial parties. First, it must be considered that the correspondences involved could have been generated in a *disjoint set scenario*. Also, there is a need for using a more robust optimisation method which is able to consider spurious mapping rejection based on some additional parameters, while conserving the consensus correspondence as a weighted mean (as much as possible). Finally, the quality of the consensus could be enhanced if the structural information of the objects that are being matched is considered. The present chapter is devoted to explain how through the use of the Bipartite (BP) framework [1] and additional definitions, the 2-CF can be enhanced to solve these issues. This upgrade will be onwards referred as the two-correspondence consensus framework based on the BP (2-CFBP).

5.2 Definitions and Notations

To facilitate the definitions of the 2-CFBP method, graph correspondences will be used. Let $G^1 = (\Sigma_v^1, \Sigma_e^1, \gamma_v^1, \gamma_e^1)$ and $G'^1 = (\Sigma'^1_v, \Sigma'^1_e, \gamma'^1_v, \gamma'^1_e)$ be two attributed graphs. For the correspondences to be bijective, after the matching process both graphs have been extended with null nodes, and thus both have an order N^1. $\Sigma_v^1 = \{v_i^1 \mid i = 1, \dots, N^1\}$ is the set of nodes and $\Sigma_e^1 = \{e_{i,j}^1 \middle| i, j \in 1, \dots, N^1\}$ is the set of edges, and $\gamma_v^1 : \Sigma_v^1 \to \Delta_v^1$ and $\gamma_e^1 : \Sigma_e^1 \to \Delta_e^1$ assign attribute values to nodes and edges in their respective domains. Coherent definitions hold for G'^1.

Given $f^1 : \Sigma_v^1 \to \Sigma'^1_v$ and $f^2 : \Sigma_v^2 \to \Sigma'^2_v$ being two correspondences from the nodes of $G^1 = (\Sigma_v^1, \Sigma_e^1, \gamma_v^1, \gamma_e^1)$ to $G'^1 = (\Sigma'^1_v, \Sigma'^1_e, \gamma'^1_v, \gamma'^1_e)$, and from the nodes of $G^2 = (\Sigma_v^2, \Sigma_e^2, \gamma_v^2, \gamma_e^2)$ to $G'^2 = (\Sigma'^2_v, \Sigma'^2_e, \gamma'^2_v, \gamma'^2_e)$ respectively (f^1 and f^2 relate nodes only to avoid edge inconsistencies), the order of G^1 and G'^1 is N^1, and the order of G^2 and G'^2 is N^2. It may happen that $N^1 \neq N^2$. In addition, it can only be assured that $\Delta_v^1 = \Delta'^1_v$ and $\Delta_v^2 = \Delta'^2_v$, yet Δ_v^1 may be different from Δ_v^2 and Δ'^1_v may be different from Δ'^2_v (similarly for the edges). This difference is in most cases related to the attributes' values rather than their structure, since it is clear that even if two different parties are intending to solve the same matching problem, they must use feature extraction methods with some sort of resemblance.

Consequentially, consider there is some level of intersection between both input node sets and also both output node sets. This intersection between graphs is modelled through functions $\zeta: \Sigma_v^1 \times \Sigma_v^2 \rightarrow \{0,1\}$ and $\zeta': \Sigma'^1_v \times \Sigma'^2_v \rightarrow \{0,1\}$. A value of $\zeta(v_i^1, v_j^1) = 1$ means that v_i^1 has to be considered the same node as v_j^2, and 0 means otherwise. Similar definitions hold for ζ'. If these functions are expressed in a matrix form, there could only be one cell with a value of 1 in each row and column, therefore the functions must only establish one intersection per node. The number of 1's in the matrix is the number of nodes that are considered the same in both graphs. For instance, if nodes mapped by two or more correspondences have attributes such as coordinate position or feature vectors, a distance threshold between these attributes can be implemented in order to associate elements from different. If the distance between the attributes of two elements from different sets is close enough, then an intersection is declared.

5.3 Methodology

As a result of this new problem definition, the 2-CFBP must aim at learning a consensus correspondence $\bar{f}^*: \Sigma_v^{1,2} \rightarrow \Sigma'^{1,2}_v$ given the four graphs G^1, G^2, G'^1 and G'^2 , the two graph correspondences f^1 and f^2, and the intersection functions ζ and ζ'. Set $\Sigma_v^{1,2}$ is composed of the union of sets Σ_v^1 and Σ_v^2, but the nodes related by function ζ are considered only once. The attributes used by the nodes contained by $\Sigma_v^{1,2}$ can be obtained by either using the minimum, maximum or average value of their respective attributes in Δ_v^1 and Δ_v^2. Coherent definitions hold for $\Sigma'^{1,2}_v$. Finally, sets $\Sigma_v^{1,2}$ and $\Sigma'^{1,2}_v$ are extended with null nodes to have both sets with an equal order $N^{1,2}$. These extended graphs will be onwards referred as $\hat{G}^1$, $\hat{G}^2$, $\hat{G}'^1$ and $\hat{G}'^2$. Also, the correspondences f^1 and f^2 are extended into $\hat{f}^1$ and $\hat{f}^2$, to make both mutually bijective. Therefore $Cost(G^1, G'^1, f^1) = Cost(\hat{G}^1, \hat{G}'^1, \hat{f}^1)$ and $Cost(G^2, G'^2, f^2) = Cost(\hat{G}^2, \hat{G}'^2, \hat{f}^2)$, and the minimisation function originally presented in equation 4.6 is now converted into

$$\bar{f}^*_\lambda = \underset{\forall f: \Sigma_v^{1,2} \rightarrow \Sigma'^{1,2}_v}{argmin} \begin{Bmatrix} (1-\lambda) \cdot \left[Cost(\hat{G}^1, \hat{G}'^1, f) + Cost(\hat{G}^2, \hat{G}'^2, f)\right] + \\ \lambda \cdot \left[\, d_H(\hat{f}^1, f) + d_H(\hat{f}^2, f)\right] \end{Bmatrix} \quad (5.1)$$

To reduce the number of variables and present synthesised results, a single weight λ which gauges the importance of both the loss function and the regularisation term will be used. Using similar properties as in the case of the 2-CF, it is possible to convert equation 5.1 into:

$$\bar{f}^*_\lambda = \underset{\forall f:\, \Sigma_v^{1,2} \to \Sigma'^{1,2}_v}{argmin} \left\{(1-\lambda)\cdot\left[\widehat{\boldsymbol{C}}^1{}_f + \widehat{\boldsymbol{C}}^2{}_f\right] + \lambda\cdot\left[\left[\mathbf{1}-\hat{F}^1\right]_f + \left[\mathbf{1}-\hat{F}^2\right]_f\right]\right\} \quad (5.2)$$

where $\widehat{\boldsymbol{C}}^1$ and $\widehat{\boldsymbol{C}}^2$ are extended cost matrices, $\hat{F}^1$ and $\hat{F}^2$ are extended correspondence matrices, and $\mathbf{1}$ is a matrix with all ones. Notice that with respect to the expression presented in the last chapter (equation 4.7), this framework considers all mappings regardless if they are known to be *a priori* similar. That is because the spurious mapping rejection property proposed in the current framework may decide that, even if both parties agree on one mapping, it could more appropriate to delete such mapping. This is also evident since the four matrices have been extended to be of size $n^{1,2} \times n^{1,2}$ cells. Nevertheless, these matrix extensions have to consider functions ζ and ζ' between nodes.

To better understand the new matrices' structure proposed in the 2-CFBP, figures 5.1.a. and 5.1.b. show the extended cost matrices $\widehat{\boldsymbol{C}}^1$ and $\widehat{\boldsymbol{C}}^2$. Rows have been split depending if the nodes belong to Σ_v^1, Σ_v^2 or both. Similarly, on the case of the columns, nodes are split according to their association to Σ'^1_v, Σ'^2_v or both. Moreover, notice that the costs of insertion and deletion ($C^1_{\varepsilon,j}$ and $C^1_{i,\varepsilon}$ for the case of $\widehat{\boldsymbol{C}}^1$, and $C^2_{\varepsilon,j}$ and $C^2_{i,\varepsilon}$ for the case of $\widehat{\boldsymbol{C}}^2$) have been added to the diagonals of the second and third quadrant of the extended cost matrix in order to perform the spurious mapping rejection. In the case of graph correspondences, these costs would directly depend on the type of node centrality that had been used for the generation of the involved graph correspondences. Likewise, figures 5.2.a. and 5.2.b. show the extended matrices $\hat{F}^1$ and $\hat{F}^2$. In this case, the first quadrants are used for indicating the inlier mappings, while the second and third quadrants are used to collocate the outlier mappings for null nodes in the input and output graphs respectively.

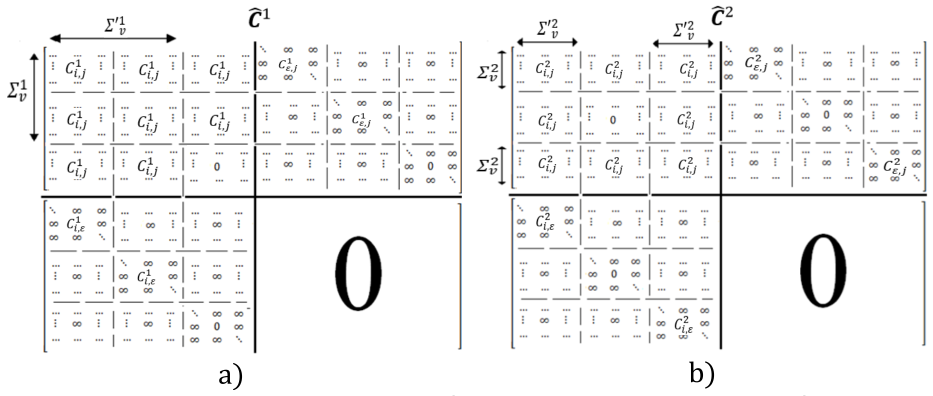


Figure 5. 1. a) Extended cost matrix $\widehat{\boldsymbol{C}}^1$. b) Extended cost matrix $\widehat{\boldsymbol{C}}^2$.

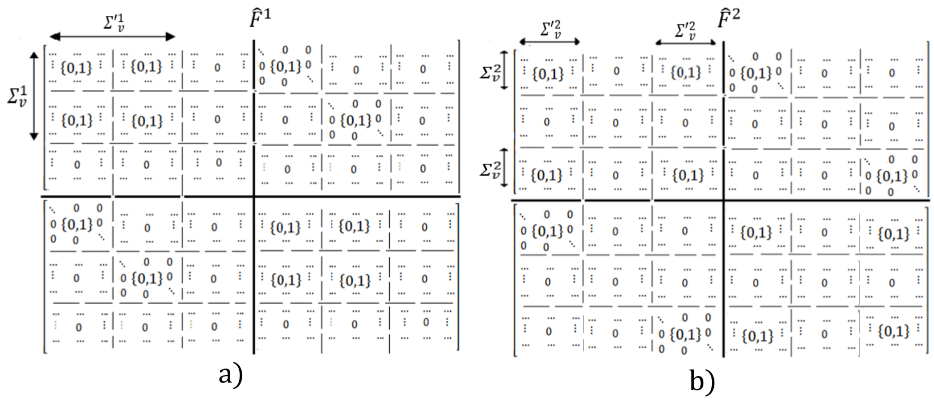


Figure 5. 2. a) Extended correspondence matrix $\hat{F}^1$. b) Extended correspondence matrix $\hat{F}^2$.

The 2-CFBP once again can be solved through the application of a LAP solver such as the Hungarian method [2], the Munkres' algorithm [3] or the Jonker-Volgenant solver [4] to learn $\bar{f}^*_\lambda$ as defined in equation 5.1, this time minimising H_λ which possesses the same structure as the matrix proposed in the BP algorithm [1].

$$H_\lambda = (1-\lambda)\cdot\left[\widehat{\boldsymbol{C}}^1+\widehat{\boldsymbol{C}}^2\right]+\lambda\cdot\left[\mathbf{2}-\left(\hat{F}^1+\hat{F}^2\right)\right] \tag{5.3}$$

where **2** is a matrix with all its values set to 2.

In the next section, we will demonstrate how the new function proposed in equation 5.3 still minimises the HD (equation 5.1) and thus, is capable to approximate the consensus correspondence to be a weighted mean correspondence (to the most).

5.4 Reasoning about Optimality

In order to show the validity of this framework, it has to be demonstrated that equation 5.1 is equal to equation 5.3. By definition of the cost, it is safe to say that $Cost(\hat{G}^1, \hat{G}'^1, f) + Cost(\hat{G}^2, \hat{G}'^2, f) = \boldsymbol{\hat{C}^1}_f + \boldsymbol{\hat{C}^2}_f$, and it is also clear that $\mathbf{1}_f = n^{1,2}$. For this reason, only the following equality must be proven to hold:

$$d_H(\hat{f}^1, f) + d_H(\hat{f}^2, f) = \mathbf{2} \cdot n^{1,2} - \hat{F}^1{}_f - \hat{F}^2{}_f \qquad (5.4)$$

Considering the relation between f, $\hat{f}^1$ and $\hat{f}^2$, the node set $\Sigma_v^{1,2}$ is split in four cases. The first two rows of table 5.1 show these four combinations. The third row shows the supposed number of nodes that hold this case (A, B, C and D represent these number of nodes). Clearly, the addition of these four values results in the total number of nodes in $\Sigma_v^{1,2}$. In the next two columns, the HD between the subsets of nodes that hold each specific case are shown. Finally, in the last two columns the obtained values of the correspondence matrices applied only to the specific nodes are shown.

Table 5.1. Four cases of nodes on $\hat{G}^1$ with respect to correspondences $\hat{f}^1$ and $\hat{f}^2$.

With respect to $\boldsymbol{\hat{f}^1}$	With respect to $\boldsymbol{\hat{f}^2}$	$\boldsymbol{\Sigma_v^{1,2}}$	$\boldsymbol{d_H(\hat{f}^1, f)}$	$\boldsymbol{d_H(\hat{f}^2, f)}$	$\boldsymbol{F^1{}_f}$	$\boldsymbol{F^2{}_f}$
$\hat{f}^1(v_i^1) \neq f(v_i^1)$	$\hat{f}^2(v_i^2) \neq f(v_i^2)$	A	A	A	0	0
$\hat{f}^1(v_i^1) \neq f(v_i^1)$	$\hat{f}^2(v_i^2) = f(v_i^2)$	B	B	0	0	B
$\hat{f}^1(v_i^1) = f(v_i^a)$	$\hat{f}^2(v_i^2) \neq f(v_i^2)$	C	0	C	C	0
$\hat{f}^1(v_i^1) = f(v_i^1)$	$\hat{f}^2(v_i^2) = f(v_i^2)$	D	0	0	D	D
TOTAL:		$\boldsymbol{n^{1,2}}$	A+B	A+C	C+D	B+D

On the one hand, taking into account the third and fourth columns $d_H(\hat{f}^1, f) + d_H(\hat{f}^2, f) = 2A + B + C$. On the other hand, taking into account the last two columns $F^1{}_f + F^2{}_f = 2D + B + C$. Therefore, $2 \cdot n^{1,2} - F^1{}_f - F^2{}_f = 2A + 2B + 2C + 2D - 2D - B - C = \boldsymbol{2A} + \boldsymbol{B} + \boldsymbol{C}$; which is exactly the same value than the one deduced for $d_H(\hat{f}^a, f^{a,b}) + d_H(\hat{f}^b, f^{a,b})$. ■

5.5 Experimental Validation

Once the database used is properly described, we propose an experiment to validate two key aspects of the present chapter. First, we aim to show that simply by using graph correspondences, the accuracy of the initial correspondences is increased compared to only using correspondences between salient point sets. Second, the experiment will intend to validate the increase in accuracy when applying the 2-CFBP.

5.5.1 Database Used

-*Tarragona Exteriors Graphs*: This database is composed of 5 sequences × 10 pairs of images × 5 node and feature extractors × 4 matching algorithms $= 1'000$ quartets of two graphs and two correspondences between them, which are split among 20 datasets (5 extractors × 4 matching algorithms). The first correspondence has been obtained with an existing matching method, and the second one is the ground truth correspondence. To construct this database, we used the 11 image samples from 5 public image databases called “BOAT”, “EASTPARK”, “EASTSOUTH”, “RESIDENCE” and “ENSIMAG” [5]. The original databases are composed of sequences of images taken of the same object, but from different camera angles and positions. Together with the images, the homography estimations that convert the first image (img00) of the set into the other ones (img01 through img10) is provided. From each of the images, we extracted the 50 most reliable salient points using 5 methodologies: FAST [6], HARRIS [7], MINEIGEN [8], SURF [9] (native Matlab 2013b libraries) and SIFT [10] (own library). Then, unattributed edges between such salient points were generated using the Delaunay triangulation method. A total of 10 correspondences between the first graph of the sequence and the other ten ones is computed through the use of four algorithms: Matlab’s *matchFeatures* function [11] (with the parameter $MaxRatio = 1$ to ensure the maximum number of inlier mappings) and the FBP method [12] applying three different node centralities: a) Node ($K_v = 0.2$) b) Degree ($K_v = K_e = 0.2$) and c) Clique ($K_v = K_e = 0.2$). The insertion and deletion costs used for the graph matching methods were set to, in such way that a fair number of inlier mappings is achieved. Finally, we construct the ground truth correspondence using the homography provided by the original image databases. In summary, the database has a total 20 datasets with a total of $1'000$ quartets (50 quartets per dataset) composed of two graphs and two correspondences $\{G_i, G'_i, f_i, h_i\}$, where $i \in [1..1'000]$. The features in the node’s feature set are the ones extracted using one of the 5 methodologies.

Notice that there are two main differences between this database and the “Tarragona Exteriors” database presented in chapter 4. The first is that graphs are extracted from the images instead of only salient points. However, notice that the key difference between the salient point data and the graph data is solely the addition of the edges, since the node features are the same than the previous dataset. Secondly, this new database offers the possibility of including more matching algorithms based on different node centralities. It is important to remark that given the edition costs used, the Node centrality with the FBP algorithm in the "Tarragona Exteriors Graphs" database delivers the same correspondence than the FBP algorithm used in the "Tarragona Exteriors" database, thus, this type of correspondence can be considered more a salient point correspondence rather than a graph correspondence. Nonetheless, the correspondences generated using either the Degree or the Clique centralities are brand new and are considered graph correspondences. This also results in having 20 datasets, instead of the original 10. Figure 5.3 shows a sample of the first graph in each sequence overlapping the image represented.

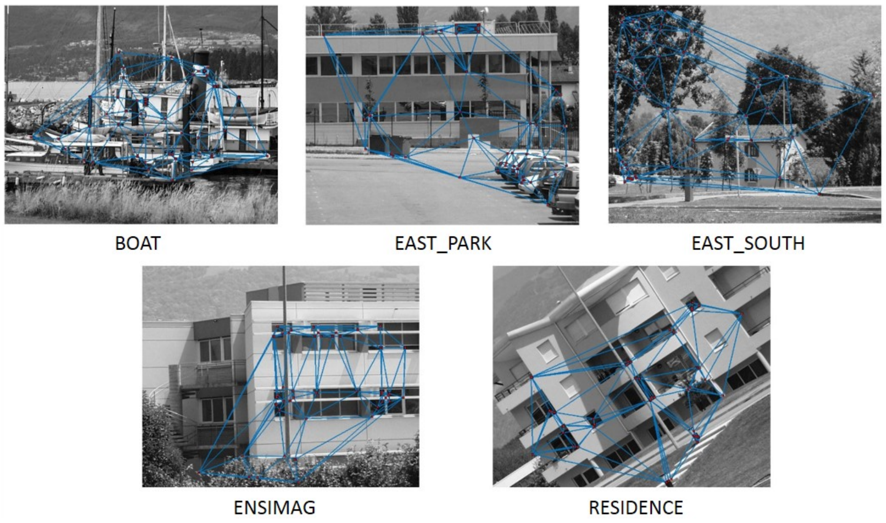


Figure 5. 3. The first graph of each sequence of the "Tarragona Exteriors" database.

The "Tarragona Exteriors Graph" database is available in [13].

5.5.2 Experimental Validation and Result Interpretation

First, this validation intends to demonstrate whether graph correspondences have more accuracy than set of point correspondences or not. If this is true and assuming the BP framework is more robust than the standard LAP solver application, then the 2-CFBP has a much higher chance of learning better consensus correspondences in terms of accuracy than the 2-CF.

Using all instances of the “Tarragona Exteriors Graphs” database, in table 5.2 we show the accuracy of each matching method (rows) in terms of the total number of correct inlier mappings found for the 10 correspondences of each sequence, sorted by the feature extractor used to detect the nodes of the graph (columns). We also include the average results for each matching algorithm (last row) and for each graph (last column). For this case, 1: FAST, 2: HARRIS, 3: MINEIGEN, 4: SURF and 5: SIFT.

Table 5.2. Results in terms of accuracy (number of total inlier mappings found) in the individual salient point correspondences datasets (*matchFeatures* and NODE) and in the individual graph correspondences datasets (DEGREE and CLIQUE).

Dataset	Name	1	2	3	4	5	Average
1	BOAT_BP-CLIQUE	62	**82**	50	47	8	**49.8**
2	BOAT_BP-DEGREE	55	58	48	38	5	40.8
3	BOAT_BP-NODE	55	62	39	32	8	39.2
4	BOAT_matchFeatures	9	7	8	65	1	18
5	EASTPARK_BP-CLIQUE	16	22	22	55	18	**26.6**
6	EASTPARK_BP-DEGREE	24	31	27	21	14	23.4
7	EASTPARK_BP-NODE	15	14	22	29	18	19.6
8	EASTPARK_matchFeatures	3	1	1	**66**	1	14.4
9	EASTSOUTH_BP-CLIQUE	8	5	7	16	6	8.4
10	EASTSOUTH_BP-DEGREE	9	5	3	24	9	**10**
11	EASTSOUTH_BP-NODE	6	6	7	14	6	7.8
12	EASTSOUTH_matchFeatures	1	1	1	**32**	5	8
13	RESIDENCE_BP-CLIQUE	22	13	20	96	6	**31.4**
14	RESIDENCE_BP-DEGREE	22	14	17	15	6	14.8
15	RESIDENCE_BP-NODE	10	16	15	13	5	11.8
16	RESIDENCE_matchFeatures	1	1	1	**106**	1	22
17	ENSIMAG_BP-CLIQUE	3	4	2	42	3	10.8
18	ENSIMAG_BP-DEGREE	3	3	3	34	2	9
19	ENSIMAG_BP-NODE	3	4	2	29	3	8.2
20	ENSIMAG_matchFeatures	1	1	1	**53**	1	**11.4**

Notice that the SURF graphs work well with the Matlab’s default matching function, and thus delivered the highest amount of inlier mappings for all sequences except for the BOAT one. Nevertheless, in the case of the other four extracted graphs, it delivered worse results. Since our main interest is the consensus of several parties of different characteristics, the qualities of this database are very proper to show that the consensus framework is useful. It is also important to remark that the BP algorithm works better when matching the FAST and HARRIS graphs, and that the sequence that delivered the highest total number of correct inlier mappings for all graphs is BOAT.

Table 5.3. Results in terms of accuracy (number of total inlier mappings found) in the consensus correspondence between salient point correspondences datasets and graph correspondences datasets.

Dataset	$E_{1,2}$	$E_{1,3}$	$E_{1,4}$	$E_{1,5}$	$E_{2,3}$	$E_{2,4}$	$E_{2,5}$	$E_{3,4}$	$E_{3,5}$	$E_{4,5}$	Average
1	84	78	104	70	96	116	89	**128**	57	80	**90.2**
2	74	72	82	60	68	109	61	111	52	42	73.1
3	70	65	84	63	56	89	70	62	42	37	63.8
4	10	10	48	9	9	50	9	46	9	66	26.6
5	35	27	84	36	22	**94**	3	83	41	85	**55**
6	29	27	31	27	33	42	42	35	40	32	33.8
7	11	23	43	35	18	38	32	46	37	46	32.9
8	2	1	38	4	2	47	1	44	2	67	20.8
9	8	19	20	12	16	27	11	27	10	30	18
10	11	7	**34**	18	3	29	14	25	12	30	**18.3**
11	9	2	18	10	4	17	10	19	12	18	11.9
12	1	1	22	1	1	20	1	22	1	32	10.2
13	32	24	124	27	25	**129**	26	122	23	100	**63.2**
14	16	19	36	26	12	26	20	29	21	21	22.6
15	10	15	24	15	12	24	16	24	19	17	17.6
16	1	1	59	1	2	51	1	39	1	106	26.2
17	4	3	47	4	5	44	6	43	5	52	**21.3**
18	4	5	34	3	1	37	5	35	3	35	16.2
19	4	5	34	3	1	37	5	35	3	35	16.2
20	1	1	42	1	1	38	1	36	1	**53**	17.5
Average	3	2.8	41.8	3.2	3	41.2	2.6	37.4	2.8	64.8	
	20.8	22	40.6	25.2	18.2	41	26.6	37.2	22.6	30.6	
	26.8	26	43.4	26.8	23.4	48.6	28.4	47	25.6	32	
	32.6	30.2	75.8	29.8	32.8	82	27	80.6	27.2	69.4	

Table 5.3 shows accuracy in terms of total number of inlier mappings found for consensus of $E_{a,b}$, represented by the combination of feature extractors a and b. Only the configuration that delivered the best results is shown. Since C_{vs} is the normalised Euclidean distance and $C_{es} = 0$ due to edges not having attributes, the following edit costs in the extended cost matrices are set: Node: $K_v = 50$, Degree: $K_v = K_e = 50$ and Clique: $K_v = K_e = 250$. From these costs, we are able to generate $C^1_{\varepsilon,j}$ and $C^1_{i,\varepsilon}$ for the case of $\widehat{\boldsymbol{C}}^1$, and $C^2_{\varepsilon,j}$ and $C^2_{i,\varepsilon}$ for the case of $\widehat{\boldsymbol{C}}^2$. Finally, the weighting parameter is set on all experiments to $\lambda = 0.2$.

Several observations can be drawn from comparing table 5.2 and 5.3. For instance, in table 5.3 for the BOAT sequence, the consensus using Clique with $E_{1,2}$ obtains 84 correct inlier mappings, given that in table 5.2, graphs extracted with 1 (FAST) mapped with FBP Clique obtained 62 correct inlier mappings and graphs extracted with 2 (HARRIS) mapped with FBP Clique obtained 82 correct inlier mappings. This does not seem much of an improvement. However, this has been caused by the fact that both involved correspondences had several intersections (both in the graph nodes and the mappings themselves) and the consensus was not able to improve much. In contrast, some other results indeed show improvement. Once again in the BOAT sequence, the consensus using Clique with $E_{4,5}$ delivers 80 correct inlier mappings, given that graphs extracted with 4 (SURF) mapped with FBP Clique obtained 47 correct inlier mappings and graphs extracted with 5 (SIFT) mapped with FBP Clique obtained 8 correct inlier mappings. This is a very considerable improvement with respect to the same sequence and graphs in the POINTS consensus, which delivered 66 correct inlier mappings, starting from 65 correct ones in the case of graph 4 (SURF) mapped with *matchFeatures*, and 1 correct one in the graph 5 (SIFT) mapped with *matchFeatures*. In the average column of the results, it is observed that for most of cases the consensus using the Clique node centrality obtains the highest amount of correct inlier mappings, except for the EASTSOUTH sequence where the Degree node centrality performs slightly better. Similarly, in the average row we observe that for all combination of graphs, the best option is the framework currently presented and using the Clique structure.

To summarise results, table 5.4 is presented. In the case of the columns labelled as “Total”, we show the combination of feature extractor to generate a graph and matching algorithm which obtained the highest number of total correct inlier mappings per sequence. In the column labelled “Average” for the individual correspondences (third column) we show the graph matching method which delivered the highest number of correct inlier mappings in average for all the extracted graphs. Finally, the column labelled “Average” for the consensus correspondences (fifth column) shows the best option to obtain a consensus correspondence in the proposed accuracy.

Table 5.4. Summary of the obtained results

Dataset	Individual Correspondences		Consensus Correspondences	
	Total	Average	Total	Average
BOAT	HARRIS-FBP CLIQUE	FBP CLIQUE	SIFT/SURF-CLIQUE	CLIQUE
EASTPARK	SURF-MATLAB	FBP CLIQUE	HARRIS/SURF-CLIQUE	CLIQUE
EASTSOUTH	SURF-MATLAB	FBP DEGREE	FAST/SURF-CLIQUE	DEGREE
RESIDENCE	SURF-MATLAB	FBP CLIQUE	HARRIS/SURF-CLIQUE	CLIQUE
ENSIMAG	SURF-MATLAB	MATLAB	SIFT/SURF-MATLAB	CLIQUE

5.6 Discussion

In this chapter, a new consensus framework called 2-CFBP has been presented, which offers several advantages with respect to the previous approach. This new proposal is designed to learn the consensus given two correspondences between objects with structural information (such as graphs), which have been generated by separate entities, and thus, present some distinct characteristics. Also, the new framework considers spurious mapping rejection through a generalisation of the BP framework [1]. Although the method has only been exemplified for its use with graph correspondences, it can also be applied for any type of correspondences by properly setting the costs used.

In the experimental section, three key aspects of this methodology have been validated. First, it has been demonstrated that innately graph matching is more accurate than salient point matching, producing correspondences that are able to exploit structural information in a better form. Second, we have shown that the 2-CFBP is capable of learning a consensus correspondence which is superior in accuracy than the involved graph correspondence proposals. As a final observation, it is shown that the node centrality used plays an important role in the accuracy result, and thus the benefit of considering different node centralities in this framework.

The next logical step, which will be presented in the following section, is to develop the multiple-input consensus framework. In this subsequent phase, it is imperative to maintain the relevant attributes that the frameworks presented so far have proven to be effective, such as the cost and distance minimisation, the use of the BP framework, the consideration of either salient point correspondences or graph correspondences and the structural information conveyed on them, and most importantly the capability to work in the disjoint set scenario; an aspect that carries special relevance given the fact that the higher the number of involved correspondences, the bigger the chance to encounter a more complex disjoint set scenario.

Chapter 6 Consensus of Multiple Correspondences

6.1 Introduction

To understand the challenges of defining a multiple-consensus framework (M-CF), consider an image registration scenario. Figure 6.1 shows two images in which four different parties identify different sets of points and correspondences.

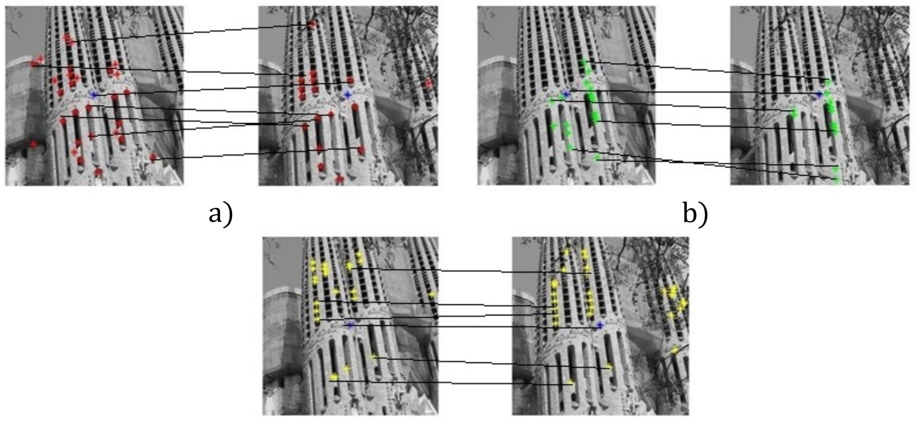


Figure 6.1. Three possible correspondences using a) extractor 1 and matching algorithm 1, b) extractor 2 and matching algorithm 2 and c) extractor 3 and matching algorithm 3.

The first issue is that if correspondences are mutually bijective, then an enormous quantity of null elements may be needed to accomplish this and thus the computation of the consensus correspondence would increase. Moreover, the intersection functions ζ and ζ' defined in the previous chapter must be able to cope with more than two sets of features, and must be able to generate properly the union sets used by the final consensus correspondence. Finally, it must be pointed out that since more than two parties are involved, then the requirement of the consensus solution to be a weighted mean now becomes obsolete. Therefore, the loss function in this case must be redefined.

There are several forms in which a M-CF can be modelled. An intuitive first solution is to adapt a voting strategy by using the GM computation framework presented in chapter 3. For that aim, it would only be necessary to consider that the input in this case is a disjoint set scenario. Another possibility is to apply the 2-CFBP (chapter 5) in an iterative manner, so that a consensus correspondence is produced once that all of the possibilities are included. Finally, a third possibility could be defining a new methodology for the problem at hand, which considers all input correspondences in an agglomerative way.

6.2 Definitions and Notations

Let $f^1 : G^1 \times G'^1$ and $f^2 : G^2 \ x \ G'^2$ be two correspondence functions between two output objects $G^1 = \{a_1^1, a_2^1, \ldots, a_{N^1}^1\}$ and $G^2 = \{a_1^2, a_2^2, \ldots, a_{N^2}^2\}$ and two input objects $G'^1 = \{a'^1_1, a'^1_2, \ldots, a'^1_{N^1}\}$ and $G'^2 = \{a'^2_1, a'^2_2, \ldots, a'^2_{N^2}\}$. Due to an extension with null elements, the order of G^1 and G'^1 is N^1, the order of G^2 and G'^2 is N^2, and therefore correspondences are bijective. The problem solved in the last chapter was to define a consensus correspondence $\bar{f}^* : G \times G'$. Then, the domain of G had been defined as $G = G^1 \cup G^2$, and the codomain as $G' = G'^1 \cup G'^2$. Therefore, the order of G and G' results in N. This solution will onwards be referred using the function $(\bar{f}^*, G, G') = PairConsensus(\lambda, f^1, G^1, G'^1, f^2, G^2, G'^2)$.

Suppose there is a set of output objects $\{G^1, \ldots, G^k, \ldots, G^M\}$ and a set of input objects $\{G'^1, \ldots, G'^k, \ldots, G'^M\}$. Each object is composed of $G^k = \{a_1^k, a_2^k, \ldots, a_{n^k}^k\}$ and $G'^k = \{a'^k_1, a'^k_2, \ldots, a'^k_{n^k}\}$. Moreover, there are $\{f^1, \ldots, f^k, \ldots, f^M\}$ correspondences, where $f^k : G^k \times G'^k$. The paired objects G^k and G'^k have the same order N^k but in general, non-paired objects have different orders. Thus, a cost function $c(a_i^k, a'^k_j)$ between elements in objects G^k and G'^k can be defined for each of these pairs. An element can be represented by different features and so, it can be included one or more objects of $\{G^1, \ldots, G^k, \ldots, G^M\}$ at the same time. For this reason, the definition of the extended object $\hat{G} = \bigcup_{k=1}^M G^k$ is possible. This condition holds equivalently for objects in $\{G'^1, \ldots, G'^k, \ldots G'^M\}$ and the union object $\hat{G}' = \bigcup_{k=1}^M G^{k\prime}$. Object $\hat{G}$ is composed of $\hat{G} = \{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{\hat{N}}\}$, where the domain of each element $\hat{a}_i$ is a vector of M elements $\hat{a}_i^1, \ldots, \hat{a}_i^k, \ldots, a_i^M$. Element $\hat{a}_i^k$ takes value a_s^k if the i^{th} element in $\hat{G}$ is considered to be the s^{th} element in G^k or takes a special null value Φ representing that this element does not exist in G^k. The same holds for G' and G'^k. Figure 6.2 shows an example composed of objects G^1, G^2 and G^3 and of objects G'^1, G'^2 and G'^3, with correspondences f^1, f^2 and f^3 between them respectively. The order of these objects is $N^1 = 2$, $N^2 = 2$ and $N^3 = 1$. Columns represent different objects, and rows represent elements between different objects, but that have to be considered the same element in the union object.

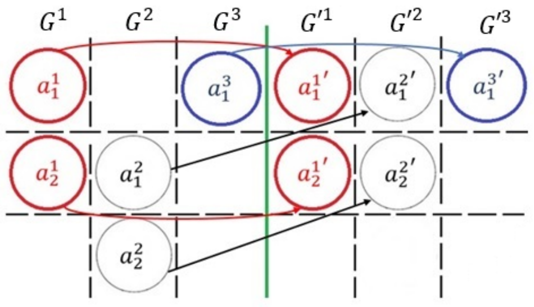
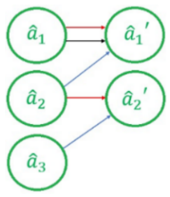


Figure 6. 2. Input of our problem composed of three pairs of objects and three correspondences between them.

Figure 6.3 shows the resulting union objects $\hat{G}$ and $\hat{G}'$ and the value of the elements' attributes



$$\hat{a}_1 = [a_1^1, \phi, a_1^3] \qquad \hat{a}_1{}' = [a_1^{1'}, a_1^{2'}, a_1^{3'}]$$

$$\hat{a}_2 = [a_2^1, a_1^2, \phi] \qquad \hat{a}_2{}' = [a_2^{1'}, a_2^{2'}, \phi]$$

$$\hat{a}_3 = [\phi, a_2^2, \phi]$$

Figure 6. 3. Example of the union set construction

The new loss function is defined as the M distances between the initial correspondences f^k proposed by the different entities and the consensus correspondence.

$$\nabla(f) = \left[\, d_H(f^1, f), \dots,\, d_H\big(f^k, f\big), \dots,\, d_H(f^M, f)\right] \tag{6.1}$$

In parallel, the regularisation term is defined as,

$$\Omega(f) = \left[Cost^1\big(\hat{G}, \hat{G}', f^1\big), \dots, Cost^k\big(\hat{G}, \hat{G}', f^k\big), \dots, Cost^M\big(\hat{G}, \hat{G}', f^M\big)\right] \tag{6.2}$$

Function $Cost^k$ computes the cost between objects $\hat{G}$ and $\hat{G}'$ given correspondence f, but only considering the k^{th} attribute in the elements of these objects, $\hat{a}_1^k, \dots \hat{a}_N^k$ and $\hat{a}_1^{k}{}', \dots \hat{a}_N^{k}{}'$. If different cost functions are available for a single element, the best is defined to be the one that results in the correspondence that conveys in the lowest cost value.

The following practical example is proposed to illustrate the problem at hand. In figure 6.4, three parties extract different salient points and feature sets from a pair of images, and match them in a bijective manner with respect to their own extracted sets of points.

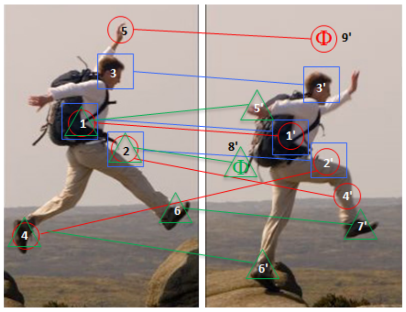


Figure 6. 4. Example for a consensus case; (⊞) f^1, (⊖) f^2, (△) f^3.

6.3 Methodology

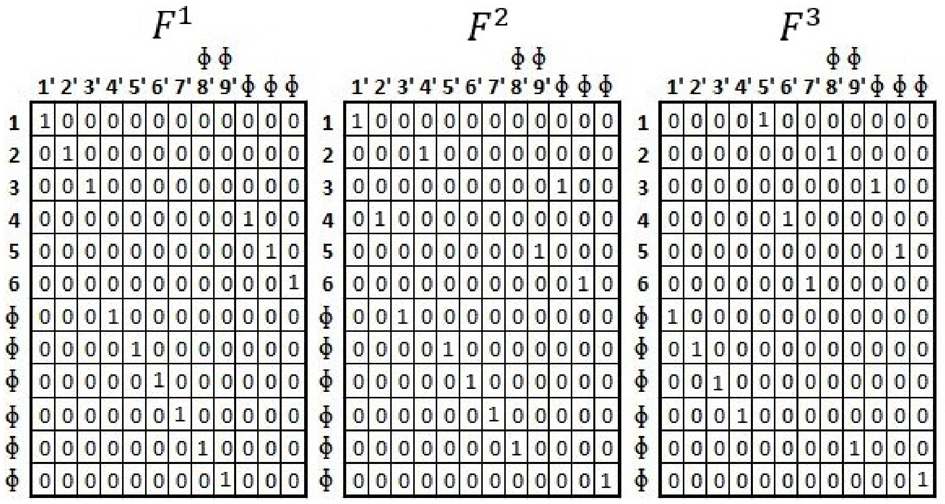
In this section, we present 3 candidate solutions to implement a M-CF. The first proposal is based on applying the voting scheme (chapter 3) with the aim of finding the GM correspondence. This solution will be onwards referred as *Voting*. The second one is based on applying the $PairConsensus$ function recursively. We have called this solution *Iterative*. Finally, in the last section we have presented some notations which will aid to define a new method that takes into account all involved sets and correspondences simultaneously using the consensus framework restrictions. This solution is named *Agglomerative*.

6.3.1 Voting Consensus Framework

Clustering or classification methods based on a voting process select as a final result the one that most of the contributions decided to be the best one. In these cases, a minimisation process is needed to arrive at the solution. Therefore, a method such as the GM calculation presented in chapter 3 could be considered a voting method.

In example provided on figure 6.4, the feature extractors obtained some points that are different between them and so, correspondences initially do not contribute on a complete solution since they do not relate the points that only appear on the other extractor. As more parties are considered, the number of discrepancies increases and so, the initial correspondences tend to be less complete.

Voting has two main steps. In the first one, the correspondence matrices $F^1, \dots, F^k, \dots, F^M$ are defined, where $F^k[i,j] = 1$ if $f^k(a_i^k) = a'^k_j$ and $F^k[i,j] = 0$ otherwise. Moreover, correspondences must be transformed into mutually bijective ones. For our practical example, this requirement has derived in correspondence matrices of size 12×12, as shown in figure 6.5.



F^1

	1'	2'	3'	4'	5'	6'	7'	ϕ 8'	ϕ 9'	ϕ	ϕ	ϕ
1	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	0	0	1
ϕ	0	0	0	1	0	0	0	0	0	0	0	0
ϕ	0	0	0	0	1	0	0	0	0	0	0	0
ϕ	0	0	0	0	0	1	0	0	0	0	0	0
ϕ	0	0	0	0	0	0	1	0	0	0	0	0
ϕ	0	0	0	0	0	0	0	1	0	0	0	0
ϕ	0	0	0	0	0	0	0	0	1	0	0	0

F^2

	1'	2'	3'	4'	5'	6'	7'	ϕ 8'	ϕ 9'	ϕ	ϕ	ϕ
1	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	0
ϕ	0	0	1	0	0	0	0	0	0	0	0	0
ϕ	0	0	0	0	1	0	0	0	0	0	0	0
ϕ	0	0	0	0	0	1	0	0	0	0	0	0
ϕ	0	0	0	0	0	0	1	0	0	0	0	0
ϕ	0	0	0	0	0	0	0	1	0	0	0	0
ϕ	0	0	0	0	0	0	0	0	0	0	0	1

F^3

	1'	2'	3'	4'	5'	6'	7'	ϕ 8'	ϕ 9'	ϕ	ϕ	ϕ
1	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	1	0	0	0	0	0
ϕ	1	0	0	0	0	0	0	0	0	0	0	0
ϕ	0	1	0	0	0	0	0	0	0	0	0	0
ϕ	0	0	1	0	0	0	0	0	0	0	0	0
ϕ	0	0	0	1	0	0	0	0	0	0	0	0
ϕ	0	0	0	0	0	0	0	0	1	0	0	0
ϕ	0	0	0	0	0	0	0	0	0	0	0	1

Figure 6. 5. Correspondence matrices F^1, F^2 and F^3 used by Voting.

In the second step, the consensus correspondence $\bar{f}^*$ is defined to minimise the following expression

$$\bar{f}^* = \underset{\forall f}{\operatorname{argmin}} \left\{ \sum_{i,j=1}^{n+m} \left(\left(\boldsymbol{M} - \sum_{k=1}^{M} F^k \right) \circ \mathrm{F} \right) [i,j] \right\} \tag{6.3}$$

where $F[i,j] = 1$ if $\bar{f}^*(a_i) = a'_j$ and $F[i,j] = 0$ otherwise. Moreover, $\circ$ represents the Hadamard product and $\boldsymbol{M}$ is a matrix with all cells with the value M.. Notice that the number of votes of a specific element mapping, for instance $a_i^k \longrightarrow a'^k_j$, is $\sum_{k=1}^{N} F^k\,[i,j]$. Therefore, searching for the correspondence that minimises $\boldsymbol{M} - \sum_{k=1}^{N} F^k$ is congruent to taking into consideration a typical voting method.

Algorithm 6.1 shows how to obtain a consensus correspondence through Voting.

Algorithm 6. 1. Voting Consensus Framework

```
Input: ${f^1, \dots, f^k, \dots f^M}, {G^1, \dots, G^k, \dots, G^M}, {G'^1, \dots, G'^k, \dots, G'^M}$
Output: $\bar{f}^*, G, G'$
Begin
for $i, j, k$
        $F^k[i,j] = 1$ if if $f^k(a_i^k) = a'^k_j$
        $F^1[i,j] = 0$ otherwise.
end for
$\bar{f}^* = LinearSolver(\boldsymbol{M} - \sum_{k=1}^{M} F^k)$
End algorithm
```

6.3.2 Iterative Consensus Framework

Algorithm 6.2 learns the consensus correspondence in an iterative manner by executing $M-1$ times the 2-CFBP that has been summarised as $PairConsensus$. Notice that this function returns the domain and codomain sets together with the correspondence so far considered the consensus one. This is because sets are extended to assure that in each execution the input correspondences are mutually bijective, and then the function produces a consensus correspondence which is bijective within the newly created set. This means that each time the function is called, the size of the sets is increased to meet the mutually bijective criteria, and thus the size of the resulting sets and correspondences increases as well. In fact, depending on the setting of the outlier costs of the 2-CFBP, more null mappings could be found and thus more elements would be added in each iteration, increasing the size of the correspondences in comparison to the former approach. In addition, the order in which correspondences are chosen has a direct repercussion on the final result, as it is usual for iterative approaches. Also, this approach has the capability of inserting different weight parameters $\{\lambda^1, \dots, \lambda^k, \dots, \lambda^M\}$ in each iteration.

Algorithm 6. 2. Iterative Consensus Framework

```
Input: ${\lambda^1, \dots, \lambda^k, \dots, \lambda^M} {f^1, \dots, f^k, \dots f^M}, {G^1, \dots, G^k, \dots, G^M}, {G'^1, \dots, G'^k, \dots, G'^M}$
Output: $\bar{f}^*, G, G'$
Begin
$\bar{f}^* = f^1$ $A = A^1$
$A' = A'^1$
for k=2 … N
        $(\bar{f}^*, G\ G') = PairConsensus(\lambda^k, \bar{f}^*, G, G', f^k, G^k, G'^k)$
end for
End algorithm
```

6.3.3 Agglomerative Consensus Framework

The function to be minimised in this case is

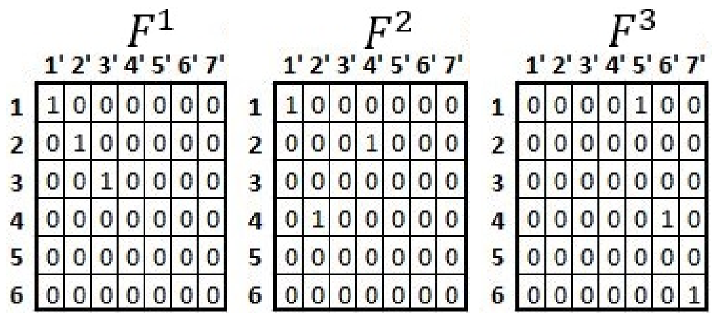
$$\bar{f}^* = \underset{\forall f}{\operatorname{argmin}} \left\{ \left[\lambda \cdot \left(\mathbf{1} - \frac{1}{M} \cdot \sum_{k=1}^{M} F^k \right) + (1-\lambda) \cdot \frac{1}{M} \cdot \sum_{k=1}^{M} C^k \right] \circ \mathrm{F} \right\} \tag{6.4}$$

Similar to the proposals of the last chapter, it is composed of a loss function $\sum_{k=1}^{M} d_H(\hat{f}^k, f)$ that has the aim of minimising the distance, together with a regularisation term $Cost(\hat{G}, \hat{G}', f)$ that has the aim of reducing the cost of the consensus. To solve the combinatorial problem of learning the consensus, the BP framework [1] is used.

In this proposal, M correspondence matrices and M cost matrices are built as follows. Both structures are composed of four quadrants. The left upper quadrant represents the set of combinations between elements in $\hat{G}$ that also belong to G^k and elements in $\hat{G}'$ that also belong to G'^k. This quadrant in F^k and C^k is defined in a similar way as the whole matrices F^k and C^k in the 2-CFBP. The second quadrant represents the combinations between elements in $\hat{G}$ that also are in G^k and elements in $\hat{G}'$ that are not in G'^k. Similarly, the third quadrant represents the combinations between elements in $\hat{G}$ that are not in G^k and elements in $\hat{G}'$ that also are in G'^k. These two last quadrants are useful to allow elements to be considered outliers. The fourth quadrant is composed of correspondences between null elements. With respect to matrix F^k, in the first, second and third quadrant, cells contain a value of 0 or 1 valued. If an element belongs to the original set G^k and is mapped, then the sum of the column or row in F^k that represents this element is 1. Otherwise, the whole column or row in F^k is 0.

In a similar way for matrix C^k, in the first quadrant, there are four different types of cells. The ones that both elements belong to G^k and G'^k, then the cost $C_{i,j}^k$ of mapping these elements are considered. If the elements do not belong to the sets G^k or G'^k, then the cost of assigning it to a null element is considered, $C_{\varepsilon,j}^k$ or $C_{i,\varepsilon}^k$. Finally, the ones where both elements do not belong to G^k or G'^k have a cost of 0. The other quadrants have been defined as the original BP framework. Nevertheless, notice there are some cells in the diagonals of the second and third quadrant that have a value of 0. These are the cases where the element does not belong to G^k (in the second quadrant) or does not belong to G'^k (in the third quadrant).

The application of this methodology on the practical example is the following. Notice the first quadrants of matrices F^1, F^2 and F^3 are filled on a non-mutually bijective form, since the second and third quadrant contain the information of the outlier mappings.



F^1

	1'	2'	3'	4'	5'	6'	7'
1	1	0	0	0	0	0	0
2	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

F^2

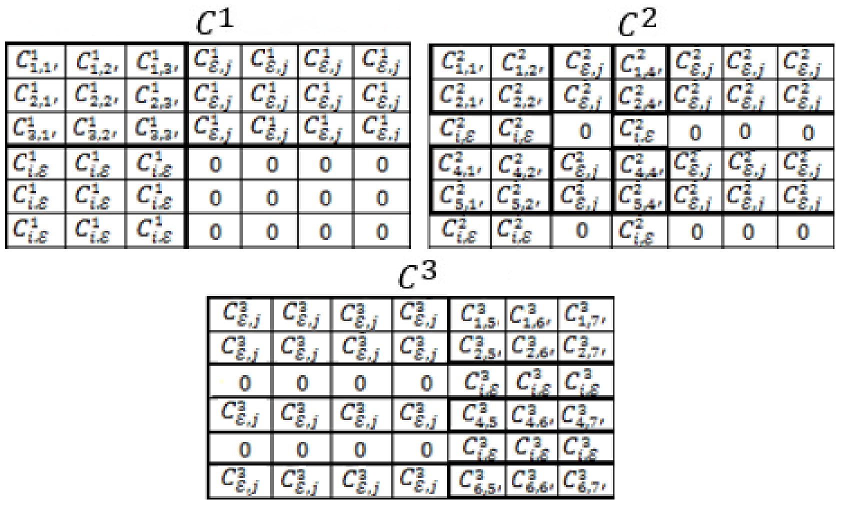
	1'	2'	3'	4'	5'	6'	7'
1	1	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

F^3

	1'	2'	3'	4'	5'	6'	7'
1	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1

Figure 6. 6. First quadrant of correspondence matrices F^1, F^2 and F^3 used in Agglomerative.

The cost matrices C^1, C^2 and C^3 are constituted as shown in figure 6.7. Note the cost of assigning an element to a null is a constant value represented by either $C^k_{\varepsilon,j}$ or $C^k_{i,\varepsilon}$.



C^1

$C^1_{1,1'}$	$C^1_{1,2'}$	$C^1_{1,3'}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$
$C^1_{2,1'}$	$C^1_{2,2'}$	$C^1_{2,3'}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$
$C^1_{3,1'}$	$C^1_{3,2'}$	$C^1_{3,3'}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,j}$	$C^1_{\varepsilon,i}$
$C^1_{i,\varepsilon}$	$C^1_{i,\varepsilon}$	$C^1_{i,\varepsilon}$	0	0	0	0
$C^1_{i,\varepsilon}$	$C^1_{i,\varepsilon}$	$C^1_{i,\varepsilon}$	0	0	0	0
$C^1_{i,\varepsilon}$	$C^1_{i,\varepsilon}$	$C^1_{i,\varepsilon}$	0	0	0	0

C^2

$C^2_{1,1'}$	$C^2_{1,2'}$	$C^2_{\varepsilon,j}$	$C^2_{1,4'}$	$C^2_{\varepsilon,j}$	$C^2_{\varepsilon,j}$	$C^2_{\varepsilon,j}$
$C^2_{2,1'}$	$C^2_{2,2'}$	$C^2_{\varepsilon,i}$	$C^2_{2,4'}$	$C^2_{\varepsilon,i}$	$C^2_{\varepsilon,i}$	$C^2_{\varepsilon,i}$
$C^2_{i,\varepsilon}$	$C^2_{i,\varepsilon}$	0	$C^2_{i,\varepsilon}$	0	0	0
$C^2_{4,1'}$	$C^2_{4,2'}$	$C^2_{\varepsilon,i}$	$C^2_{4,4'}$	$C^2_{\varepsilon,j}$	$C^2_{\varepsilon,j}$	$C^2_{\varepsilon,j}$
$C^2_{5,1'}$	$C^2_{5,2'}$	$C^2_{\varepsilon,i}$	$C^2_{5,4'}$	$C^2_{\varepsilon,i}$	$C^2_{\varepsilon,i}$	$C^2_{\varepsilon,i}$
$C^2_{i,\varepsilon}$	$C^2_{i,\varepsilon}$	0	$C^2_{i,\varepsilon}$	0	0	0

C^3

$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{1,5'}$	$C^3_{1,6'}$	$C^3_{1,7'}$
$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{2,5'}$	$C^3_{2,6'}$	$C^3_{2,7'}$
0	0	0	0	$C^3_{i,\varepsilon}$	$C^3_{i,\varepsilon}$	$C^3_{i,\varepsilon}$
$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{4,5}$	$C^3_{4,6'}$	$C^3_{4,7'}$
0	0	0	0	$C^3_{i,\varepsilon}$	$C^3_{i,\varepsilon}$	$C^3_{i,\varepsilon}$
$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{\varepsilon,j}$	$C^3_{6,5'}$	$C^3_{6,6'}$	$C^3_{6,7'}$

Figure 6.7. First quadrants of cost matrices C^1, C^2 and C^3 used in Agglomerative.

Figure 6.8 shows the final consensus correspondence. Notice that in this case, a new null element has appeared (depicted between the subject's legs on the input image). This means that the linear solver has selected one of the cells on the diagonal of the second quadrant, and thus has created a null mapping for that node. This result depended directly on the setting of weighting parameter λ and also on the outlier costs used.

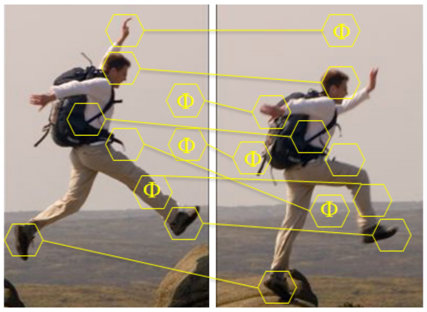


Figure 6. 8. Final correspondence for the practical example.

Algorithm 6.3 shows the computation of Agglomerative. First, matrices $F^1, \dots, F^k, \dots, F^M$ are generated such that all of them have the same number of columns and rows. Then, a function called $CostMatrix$ constructs matrices $C^1, \dots, C^k, \dots, C^M$. Finally, a linear solver is applied on the resulting matrix H.

Algorithm 6. 3. Agglomerative Consensus Framework

```
**Input:** $\{\lambda^1, \dots, \lambda^k, \dots, \lambda^M\}, \{f^1, \dots, f^k, \dots f^M\}, \{G^1, \dots, G^k, \dots, G^M\}, \{G'^1, \dots, G'^k, \dots, G'^M\}$
**Output:** $\bar{f}^*, G, G'$
**Begin**
*for* $i, j, k$
        $F^k[i,j] = 1$ $if$ if $f^k(a_i^k) = a'^k_j$
        $F^1[i,j] = 0$ $otherwise$.
*end for*
$(C^1, \dots, C^k, \dots, C^M) = CostMatrix(\{G^1, \dots, G^k, \dots, G^M\}, \{G'^1, \dots, G'^k, \dots, G'^M\})$
$H = \lambda \cdot \left(1 - \frac{1}{M} \cdot \sum_{k=1}^{M} F^k\right) + (1 - \lambda) \cdot \frac{1}{M} \cdot \sum_{k=1}^{M} C^k$
$\bar{f}^* = LinearSolver(H)$
**End algorithm**
```

6.4 Experimental Validation

Once the database used is properly described, we propose two experimental validations. In the first one, a total of 10 initial parties participate in the consensus scenario and the performance of the three proposed methods is evaluated using not only the accuracy, but also through outcomes such as the total number of mappings, the end point error and the runtime of each algorithm. This validation is performed by testing multiple initial conditions to deepen this comparison, such as the size of the input and output sets, the weight parameter λ and the edit costs used (in the case of Iterative and Agglomerative). In a second validation, we gradually increase the number of involved correspondences (from 2 until 10) and evaluate the same outcomes on the three methods. We call this a "Trade-off" Validation.

6.4.1 Database Used

-*Tarragona Exteriors* 2*:* This database is composed of 5 sequences $\times$ 10 pairs of images $\times$ 5 salient point and feature extractors $\times$ 2 matching algorithms $=$ 500 quartets (DB1), plus 7 sequences $\times$ 5 pairs of images $\times$ 5 salient point and feature extractors $\times$ 2 matching algorithms $=$ 350 quartets (DB2), which consist of two sets of salient points and two correspondences between them, which are split among 10 datasets (5 extractors $\times$ 2 matching algorithms). The first correspondence has been obtained with an existing matching method, and the second one is the ground truth correspondence. To construct this database, we used the 11 image samples from 5 public image databases called "BOAT", "EASTPARK", "EASTSOUTH", "RESIDENCE" and "ENSIMAG" (DB1) [2], as well as 7 public image databases called "BARK", "BIKES", "GRAF", "LEUVEN", "TREES", "UBC" and "WALL" (DB2) [3]. The original databases are composed of sequences of images taken of the same object, but from different camera angles and positions. Together with the images, the homography estimations that convert the first image (img00) of the set into the other ones (img01 through img10 in case of DB1 and img01 through img06 in case of DB2) is provided. From each of the images, we extracted the 75 most reliable salient points using 5 methodologies: FAST [4], HARRIS [5], MINEIGEN [6], SURF [7] (native Matlab 2013b libraries) and SIFT [8] (own library). A total of 10 correspondences between the first image of the sequence and the other ten ones is computed through the use of two algorithms: Matlab's *matchFeatures* (MF) function [9] (with the parameter $MaxRatio = 1$ to ensure the maximum number of inlier mappings) and the FBP method [10] (with an insertion and deletion cost of $K_v = 0.2$ that allowed a fair number of inlier mappings). Finally, we construct the ground truth correspondence using the homography provided by the original image databases. In summary, the database, divided into two sub-databases identified as DB1 and DB2, has a total 10 datasets with a total of 850 quartets (50 quartets per dataset) composed of two sets of features and two correspondences

$\{G_i, G'_i, f_i, h_i\}$, where $i \in [1, \dots, 500]$. The first image corresponding to each sequence in the DB2 portion are shown in figure 6.9.

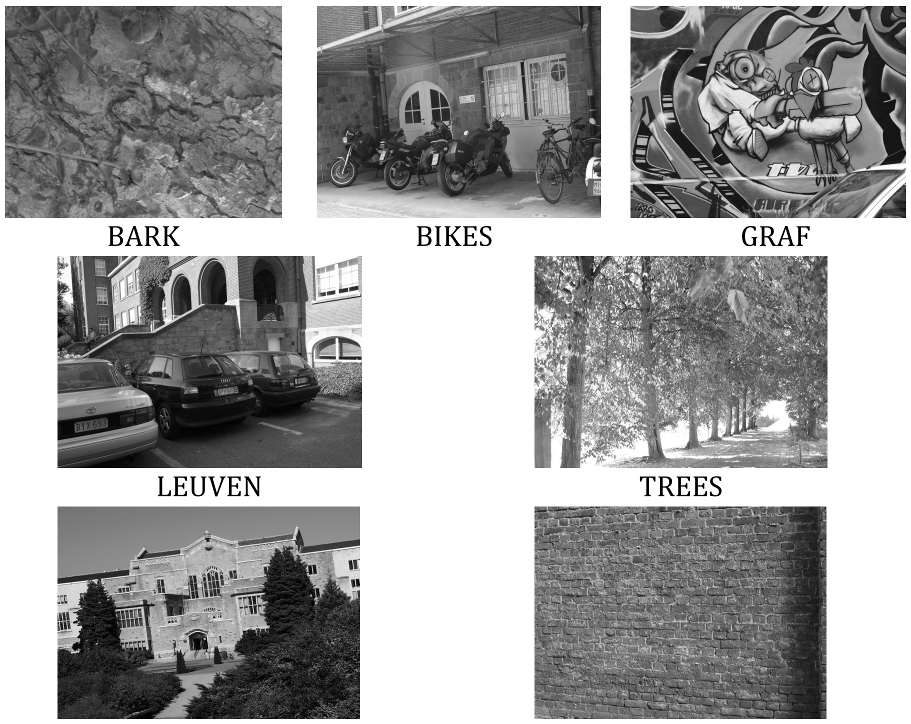


Figure 6.9. The first image corresponding to each sequence in the DB2 portion.

This database is available in [11].

6.4.2 Validation of the Three Methodologies

In this validation, only the DB1 portion of the "Tarragona Exteriors 2" database is used, given the large density of outlier mappings present in the DB2 portion. For these experiments, both Iterative and Agglomerative are run in the entire dataset a total of 54 times, considering 3 subsets s of all salient points detected in each image ($s = 25,50,75$) $\times$ 2 configurations of the outlier cost (OC) used in the extended cost matrix ($C_{\varepsilon,j} = C_{i,\varepsilon} = 0.1$ and $C_{\varepsilon,j} = C_{i,\varepsilon} = 0.5$) $\times$ 9 configurations of the gauging parameter ($\lambda = 0, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.5, 0.66, 1$). For the case of Voting, given the fact that neither the edit cost nor the λ exist, only 3 runs of the algorithm (for each the aforementioned values of s) are required. Additionally, we present the results obtained using the initial methods for the three values of s.

Finally, it must be noted that the highest values for each row will be marked in bold to identify which feature extractor-matching algorithm combination (in the case of the individual methods) or parameter configuration (in the case of the consensus frameworks) obtains the best results.

The following outcomes have been evaluated. First, to have a reference of how dense is the number of inliers for each method, this parameter is shown in tables 6.1 and 6.2. This aspect is crucial when applications demand for a large inlier density. Afterwards, we show the number of correct inliers (tables 6.3 and 6.4) and also the average cost of such inliers (tables 6.5 and 6.6). This outcome is useful to validate that the framework really tends to minimise the cost function. That is, from the obtained inliers, how similar the generated correspondences are with respect to the ground truth correspondence. Finally, the end point error has been measured (tables 6.7 and 6.8). This last outcome evidences the three frameworks from the image registration point of view, since the aim of such applications is to find the correspondence between points such that the end point error is minimised. Finally, the runtime spent to match two sets of sets (table 6.9) and to find a consensus correspondence (table 6.10) is presented. Notice that a consensus correspondence obtained through Iterative depends on the data order; in this case being the same as the order presented in the database.

-Number of Total Inlier Mappings: Table 6.1 shows the average number of inliers detected by the five feature extractors and given the two commented initial matching algorithms. We present these results to have a baseline of the inliers that each method is capable to detect. To begin, it is noticed that BP based algorithms tend to obtain more inliers than those using MF, but clearly, this fact depends on the parameterisation of these two algorithms. Also, notice that that when the matching algorithms are used in combination with the SURF feature extractor, the correspondence obtains the highest density of inlier mappings.

Table 6. 1. Number of detected inliers in the individual methods.

Individual	s	FAST	HARRIS	MINEIGEN	SURF	SIFT	Average
MF	25	556	593	595	**639**	355	547.6
	50	1137	1135	1141	**1207**	761	1706.2
	75	1683	1725	1710	**1786**	1168	1614.4
BP	25	1036	1079	1059	**1224**	1221	1123.8
	50	2132	2069	2094	**2449**	2446	2238
	75	3186	3193	3193	**3660**	3660	3378.4

Table 6.2 shows the number of inliers obtained by the consensus methods. For Agglomerative $OC = 0.1$ and the highest value of λ, the highest number of detected inliers was obtained. In $OC = 0.5$, the same number of detected inliers on any configuration of λ was obtained, except for the highest value. This is caused by the nature of the linear solver used for the consensus, which given a high OC value, tends to detect more inliers and vice versa. Separately, the average of the individual methods, Iterative (Iter) and Voting obtained fewer inliers than the best configuration of Agglomerative.

Notice that as the value of s is modified, a practically linear increase in terms of number of detected inliers either in the individual methods or in the consensus frameworks is always obtained.

Table 6. 2. Number of detected inliers in the three consensus frameworks.

Aggl	s	λ=0	0.0001	0.001	0.01	0.1	0.2	0.5	0.66	1
OC 0.1	25	3125	3125	3124	3126	3132	3130	3127	3121	**3708**
	50	6222	6221	6221	6222	6221	6218	6225	6220	**7289**
	75	9294	9294	9294	9292	9270	9267	9281	9268	**10718**
OC 0.5	25	**3878**	**3878**	**3878**	**3878**	**3878**	**3878**	**3878**	**3878**	3708
	50	**7580**	**7580**	**7580**	**7580**	**7580**	**7580**	**7580**	**7580**	7289
	75	**11240**	**11240**	**11240**	**11240**	**11240**	**11240**	**11240**	**11240**	10718
Iter	s	λ=0	0.0001	0.001	0.01	0.1	0.2	0.5	0.66	1
OC 0.1	25	1806	1807	1803	**1822**	908	273	273	273	273
	50	3686	3685	3703	**3711**	1788	573	573	573	573
	75	5568	5566	5558	**5583**	2628	792	791	791	791
OC 0.5	25	1806	1807	1803	1822	**1861**	**1861**	1173	273	273
	50	3687	3685	3702	3712	3806	**3807**	2497	574	574
	75	5570	5565	5558	5586	5691	**5694**	3786	791	791

Voting	s	
	25	3031
	50	7300
	75	10552

-Number of Correct Inliers: It has been so far observed that using Agglomerative with a high OC is the best form to obtain a large density of detected inliers, regardless of the value set for λ. Nevertheless, obtaining more or less inliers is not necessarily a good metric per se, since the aim of the presented frameworks so far has been to increase the number of correct inliers. Table 6.3 show the number of correct inliers found with each method. As previously shown in chapter 5, the SURF extractor presents the best affinity in terms of detection of correct inliers with both the MF and BP matching algorithms. Notice that if these results are compared with the ones on table 6.1, we do not see a high rate of correct inliers with respect to the detected ones. This is what makes the datasets interesting from the consensus framework point of view; since we can test the ability of the

different frameworks to deduce a bigger amount of correct inliers even if the input correspondences have a low quality.

Table 6. 3. Number of correct inliers in the individual methods.

Individual	s	FAST	HARRIS	MINEIGEN	SURF	SIFT	Average
MF	25	42	42	44	**55**	12	39
	50	56	71	60	**73**	15	55
	75	63	67	65	**73**	22	58
BP	25	69	74	63	**100**	38	68.8
	50	94	108	86	**123**	30	88.2
	75	107	117	107	**124**	39	98.8

Table 6.4 shows the number of correct inliers for the three methodologies. Once again, Agglomerative outperforms the two others. Also, selecting a value of $0.1 \leq \lambda \leq 0.2$ guarantees Agglomerative and Iterative to find the best result, effectively showing that an optimal consensus correspondence is obtained when the λ parameter is properly set, so that both terms contribute. In the case of Agglomerative, gauging the OC parameter did not represent much of a difference compared to the case of the values shown in table 6.2. Therefore, selecting for instance $OC = 0.1$ guarantees a considerably high number of correct inliers. In this sense, using different values of s does not deliver a linear increase in terms of number of correct inliers detected.

Table 6. 4. Number of correct inliers in the three consensus frameworks.

Aggl	s	ʎ=0	0.0001	0.001	0.01	0.1	0.2	0.5	0.66	1
OC 0.1	25	190	189	189	196	208	**211**	210	172	156
	50	255	255	256	255	257	**264**	258	255	253
	75	267	267	269	285	**294**	292	285	283	287
OC 0.5	25	210	210	210	218	225	**227**	234	232	220
	50	263	263	265	268	269	267	274	**276**	273
	75	282	282	284	293	**299**	298	294	289	287
Iter	**s**	**ʎ=0**	**0.0001**	**0.001**	**0.01**	**0.1**	**0.2**	**0.5**	**0.66**	**1**
OC 0.1	25	45	43	42	44	**55**	25	25	25	22
	50	53	60	62	73	**75**	35	35	35	35
	75	86	86	88	91	**92**	27	27	27	27
OC 0.5	25	45	43	42	44	77	**78**	64	23	23
	50	54	60	63	72	98	**99**	93	34	34
	75	86	86	88	89	**133**	130	112	27	27

Voting	s	
	25	83
	50	124
	75	137

-*Average Cost:* Table 6.5 shows the average cost generated through mapping the inliers over the number of correct inliers detected. Except for the SIFT (higher) and SURF (lower) cases, average costs are almost similar.

Table 6. 5. Average cost of correct inliers in the individual methods.

Individual	s	FAST	HARRIS	MINEIGEN	SURF	SIFT	Average
MF	25	0.24	0.28	0.22	**0.11**	0.67	0.31
	50	0.33	0.27	0.53	**0.15**	1.19	0.49
	75	0.48	0.51	0.48	**0.23**	1.44	0.63
BP	25	0.31	0.42	0.56	**0.11**	1.04	0.49
	50	0.55	0.7	0.6	**0.18**	2.27	0.86
	75	0.74	0.42	0.83	**0.26**	2.72	0.99

Table 6.6 shows the same outcome for the consensus frameworks (lower values are put in bold). In Agglomerative, the best combinations of parameters is $\lambda = 0.2$ and $OC = 0.1$. This is the same approximate value for parameter λ in which Agglomerative (table 6.4) presented the highest accuracy. Also, notice that this achieved cost is clearly lower than the average cost obtained by the individual methods. Although Iterative does not perform as good as Agglomerative, it is still capable of minimising the cost of the consensus correspondence in high values of λ. Nevertheless, in these cases (as shown in table 6.4), accuracy is also reduced. Finally, Voting obtains a larger cost than the average value and only performs better than the individual methods using SIFT.

Only for some instances, it outperforms Iterative. Adding the fact that the cost increase is not significant as s is increased, these results validate that through the consensus correspondence, the impact of the noise is reduced and so, the quality of the mappings increases.

Table 6. 6. Average normalised cost of correct inliers in the three consensus methods.

Aggl	s	ʎ=0	0.0001	0.001	0.01	0.1	0.2	0.5	0.66	1
OC 0.1	25	0.18	0.18	0.18	0.18	0.17	**0.16**	0.17	0.2	0.34
	50	0.25	0.25	0.25	0.26	**0.24**	0.26	0.27	0.27	0.4
	75	0.34	0.34	0.34	**0.33**	**0.33**	0.34	0.35	0.35	0.54
OC 0.5	25	0.29	0.29	0.29	0.28	0.26	**0.25**	0.26	0.27	0.26
	50	0.41	0.41	0.41	**0.39**	0.41	0.42	0.42	0.42	0.4
	75	0.55	0.55	0.54	**0.53**	**0.53**	0.54	0.55	0.56	0.54
Iter	s	ʎ=0	0.0001	0.001	0.01	0.1	0.2	0.5	0.66	1
OC 0.1	25	0.69	0.7	0.71	0.53	0.38	**0.2**	**0.2**	**0.2**	0.21
	50	0.65	0.53	0.53	0.49	0.44	**0.32**	**0.32**	**0.32**	0.35
	75	0.71	0.72	0.91	0.53	**0.46**	0.62	0.62	0.62	0.62
OC 0.5	25	0.69	0.7	0.71	0.53	0.5	0.44	0.44	0.39	**0.2**
	50	0.65	0.53	0.53	0.49	0.58	0.57	0.54	**0.42**	**0.42**
	75	0.71	0.72	0.71	0.94	0.67	0.7	0.68	**0.62**	**0.62**

Voting	s	
	25	0.29
	50	0.61
	75	0.79

-*Average Normalised End Point Error:* The end point error is useful to show how a method performs when applied to image registration. A low value implies that the obtained correspondence is close to the ground truth. Table 6.7 shows the end point error over the correctly detected inliers. The correspondences generated through the MF algorithm obtained almost twice the end point error with respect to the ones using the FBP algorithm.

Table 6.7. Average cost of correct inliers in the individual methods.

Individual	s	FAST	HARRIS	MINEIGEN	SURF	SIFT	Average
MF	25	1151	1207	1061	**249**	2427	1219.36
	50	505	662	1125	**181**	2000	895.21
	75	886	928	863	**191**	1454	864.9
BP	25	903	989	1212	**115**	650	774.17
	50	816	815	847	**95**	737	662.4
	75	788	356	825	**96**	604	533.89

Table 6.8 shows the same outcome for the consensus proposals (lowest values are put in bold). The behaviour of this metric is quite constant among the tested values of λ and OC. When $s = 25$ in Agglomerative, the minimum value is smaller than the lowest of the individual methods (SURF-BP), and it is 16 times better than the average value of table 6.7. Using Iterative, we notice that between $s = 50$ and $s = 75$ the values increase in some cases. However, when comparing all of these values to the ones obtained by Agglomerative, we observe that Iterative obtained worse values; approximately 10 times larger than the ones obtained by Agglomerative. Meanwhile, Voting behaves slightly worse than Agglomerative in this sense.

Table 6.8. Average normalised end point error in the three consensus methods.

Aggl	s	λ=0	0.0001	0.001	0.01	0.1	0.2	0.5	0.66	1
OC 0.1	25	88	90	90	87	77	**75**	77	90	104
	50	64	65	64	62	64	**60**	65	65	65
	75	58	58	58	58	**57**	**57**	58	59	60
OC 0.5	25	84	85	85	83	77	**75**	76	77	78
	50	65	66	65	63	66	**62**	66	66	65
	75	59	60	59	58	58	**57**	58	60	60
Iter	s	λ=0	0.0001	0.001	0.01	0.1	0.2	0.5	0.66	1
OC 0.1	25	920	928	936	**721**	968	1371	1372	1372	1377
	50	442	362	367	**340**	510	1167	1168	1167	1167
	75	344	347	453	**243**	349	1346	1347	1347	1346
OC 0.5	25	920	928	936	721	403	**402**	895	1372	1371
	50	441	362	366	**342**	353	351	478	1168	1167
	75	344	347	453	244	**169**	181	321	1347	1347

Voting	s	
	25	90
	50	101
	75	87

The most important observation for this metric is the fact that except for Voting, when the number of salient points involved increases, the end point error is reduced. This indicates that when the number of salient points used is increased, the obtained consensus correspondence becomes closer to the optimal one.

-Runtime: Table 6.9 shows the average runtime in seconds spent to generate the correspondence between two sets of elements that represent images in the datasets. The time to extract the salient points was not considered for this runtime. Tests were performed using a PC with Intel 3.4 GHz CPU and Windows 7 operating system. Notice that correspondences using the MF algorithm are generated faster than the FBP ones, regardless of the feature extractor used. Moreover, as the number of salient points is increased, this runtime difference increases as well. However, this increase is linear for MF, but polynomial for FBP. This is evidently due to the nature of both methods; MF is a linear comparer, while the FBP algorithm has a $O(n^3)$ complexity.

Table 6. 9. Average runtime (seconds) in the individual correspondences.

Individual	s	FAST	HARRIS	MINEIGEN	SURF	SIFT	Average
MF	25	0.003	**0.001**	**0.001**	**0.001**	**0.001**	0.003
	50	0.004	0.002	**0.001**	0.002	**0.001**	0.004
	75	0.006	**0.001**	**0.001**	0.002	**0.001**	0.006
BP	25	**0.021**	**0.021**	**0.021**	0.035	0.025	**0.021**
	50	0.164	**0.158**	0.161	0.278	0.183	0.164
	75	0.433	0.427	**0.419**	0.775	0.451	0.433

For the runtime of the three consensus proposals, a different table scheme than the previous one is used in table 6.10. It is more convenient to show the runtime for each sequence in the DB1 portion of the database, and then to display the average runtime to produce one consensus correspondence for each method. That is because changing parameters λ and OC does not have a considerable impact in the runtime. Given the large number of outlier mappings obtained when using the "ENSIMAG" sequence, the intersections between inlier elements and mappings are smaller and thus, the consensus frameworks require less computational time to generate a result. In general, notice Voting is faster due to the smaller matrix sizes and the lack of a cost matrix generation. In the case of Iterative, the runtime shown corresponds only to one permutation of data input.

Table 6. 10. Average runtime (seconds) to find a consensus correspondence.

	s	BOAT	EASTPARK	EASTSOUTH	RESIDENCE	ENSIMAG	Average
Agglomerative	25	9.89	9.3	9.1	9.2	8.98	9.294
	50	166.09	156.7	151.44	148.51	140.09	152.566
	75	594.89	569.48	547.74	479.13	486.12	535.472
Iterative	25	36.83	30.26	31.77	28.41	22.73	30
	50	550.73	454.78	416.32	415.64	277.47	422.988
	75	2651.7	2039.6	2035.1	2031.2	1228.1	1997.14
Voting	25	9.13	8.64	8.4	7.94	7.348	8.2916
	50	95.18	86.79	79.13	79.78	68.01	81.778
	75	442.81	432.85	403.1	409.58	332.03	404.074

As a conclusion for this experimental validation, we have noticed that Agglomerative obtains the highest number of correct inliers, the lowest average normalised cost and the lowest end point error. Although it is slightly slower than Voting, we consider that it is the most convenient methodology to use out of the three ones explored. However, it is possible that all data is not available all at once; it would be in this case that the Iterative framework could be used in a similar form as an online learning process.

6.4.3 Trade-off Validation

For this validation, both instances DB1 and DB2 from the "Tarragona Exteriors 2" database were used. It was of particular interest to analyse whether by including less input individual correspondences, the methods could have a significant decrease of success in terms of the previously evaluated outcomes. Therefore, we executed the three consensus methodologies testing separately from $M = 2$ until $M = 10$ involved individual correspondences. Correspondences were chosen in an order according to their accuracy, (table 6.3), beginning with the two that obtained the least accuracy, until all had been involved. For Agglomerative and Iterative, the parameters were set to $\lambda = 0.2$ and $OC = 0.5$; which were the values that delivered the best results on the previous validation. The number of default salient points used was set to $s = 50$.

Figures 6.10 to 6.14 show the five previously analysed variables as the number of included initial correspondences varies. First, figure 6.11 shows that the number of detected correspondences always increases as the number of initial correspondences does. This is a natural output of any consensus framework which, by including more initial inputs, generates more correspondences.

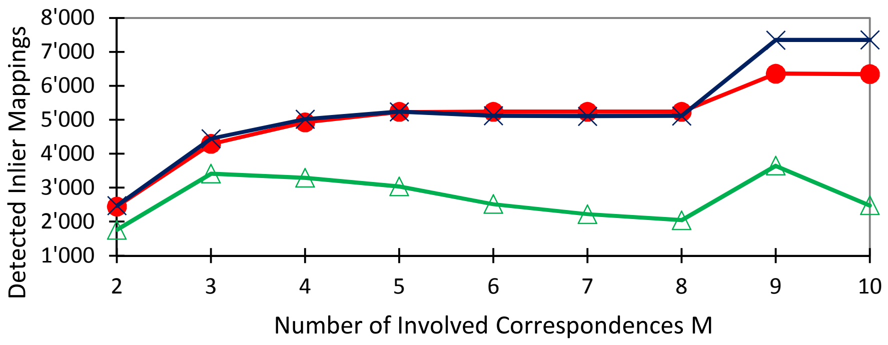


Figure 6. 10. Number of detected inlier mappings as the number of involved individual correspondences increase; (●) Agglomerative, (×) Voting, (Δ) Iterative.

The most interesting observations derive from the outcome of figure 6.11, where it is clearly noticeable that Iterative fails to increase the rate of correct inlier mappings when $M > 2$. This leads to discard Iterative as a viable option, since adding more initial correspondences reduces its efficiency. Conversely, both Agglomerative and Voting increase steadily as more inputs are added. Nonetheless, it is in Agglomerative where more correct inlier mappings are obtained.

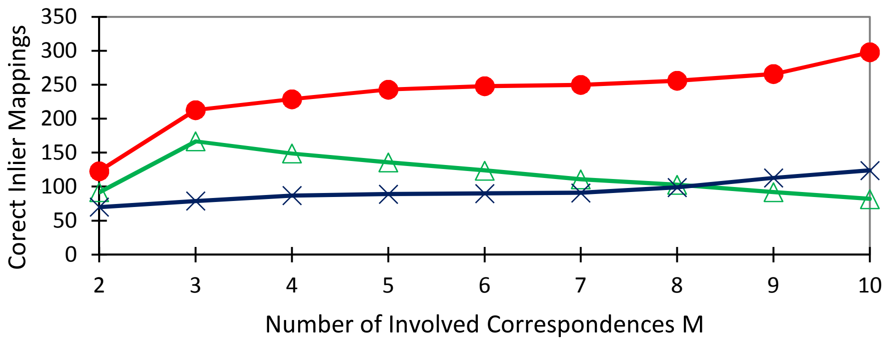


Figure 6. 11. Number of correct inliers as the number of involved individual correspondences increase; (●) Agglomerative, (×) Voting, (Δ) Iterative.

Figures 6.12 and 6.13 show the average cost and end point error respectively. It can be seen that Agglomerative is the option which in general minimises both outcomes best.

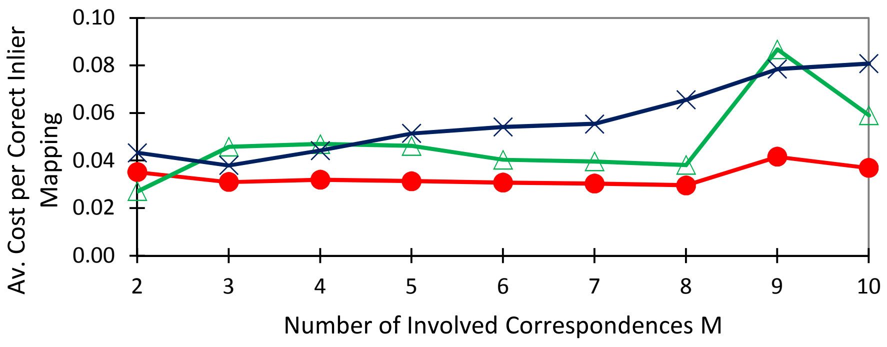


Figure 6. 12. Average cost per inlier mapping as the number of involved individual correspondences increase; (●) Agglomerative, (×) Voting, (Δ) Iterative.

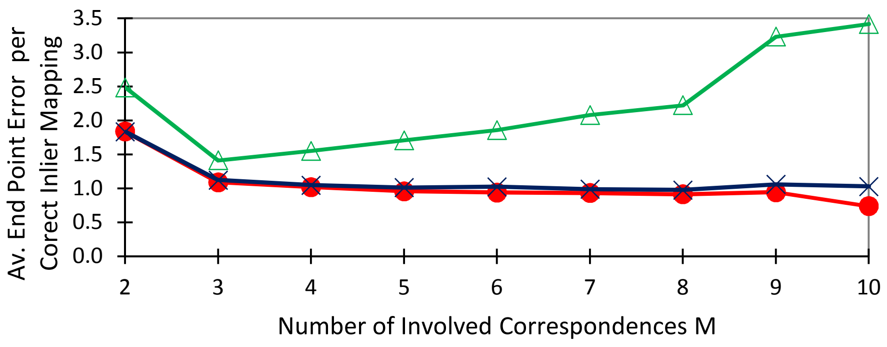


Figure 6. 13. Average End Point Error per inlier mapping as the number of involved individual correspondences increase; (●) Agglomerative, (×) Voting, (Δ) Iterative.

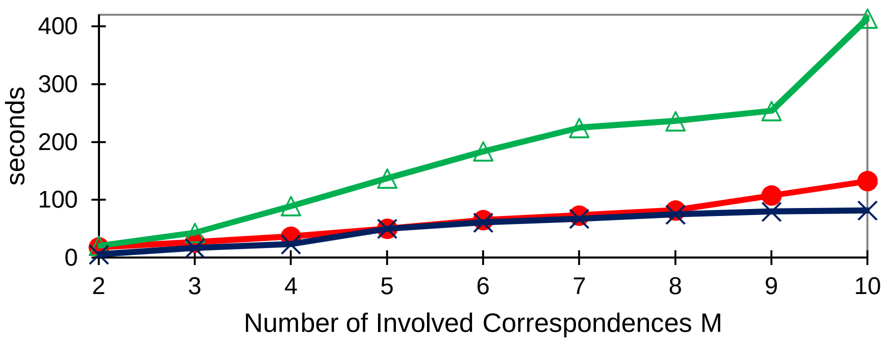


Figure 6. 14. Runtime (in seconds) as the number of involved individual correspondences increase; (●) Agglomerative, (×) Voting, (Δ) Iterative.

Finally, figure 6.14 shows that there is a considerable runtime difference between using Iterative and the other two approaches. Firstly, the increasing iteration performed on Iterative make this approach the slowest regardless of the matrix size. Also, Voting is slightly faster than Agglomerative as more inputs are used, since the matrix size are smaller and the cost matrix calculation is omitted. In the specific case of Agglomerative, 18 seconds are spent when computing a consensus between two individual correspondences, while it takes the method 152 seconds when 10 correspondences are used. This runtime difference is significant, considering that the number of correct correspondences was doubled; from 128 (when $M = 2$) to 298 (when $M = 10$). According to the system needs, it is essential to always prioritise either accuracy or runtime, and as shown in this validation, consensus frameworks are no exception.

6.5 Discussion

Three options have been explored in this chapter to clearly understand the advantages and disadvantages of each approach before making a selection for a M-CF. Several outcomes have been evaluated, such as the number of total inlier mappings found, the cost, the end point error, and the runtime. Moreover, this experimental validation has been useful not only to identify the best methodology, but to define the best parameter configuration that must be set for other tests. Additionally, a "Trade-off Validation" has been executed to verify that the selected methodology indeed gains accuracy as the number of involved correspondences increases while compensating in runtime and cost effectiveness.

Furthermore, the M-CF could include the structural information of the objects matched, such as the case of graph correspondences. We have omitted this aspect from this chapter given the large quantity of information that had to be presented. Yet, the definitions and notations for correspondences where structural information must be considered will be retaken on the next chapter. Furthermore, an important feature will be added to the framework as well; the ability of learning the weights of the involved correspondences. That is, if there is information at hand regarding the quality or confidence of the involved parties, then it is possible find an even more suitable consensus correspondence.

Chapter 7 Learning Weights for the Consensus of Multiple Correspondences

7.1 Introduction

In the previous chapter, it has been defined based on the study of several outcomes that an agglomerative approach for the M-CF is adequate for learning the best possible consensus correspondences, having the best trade-off between accuracy and runtime. Using this approach, it is possible to include all the required constraints of the original model, while simultaneously avoiding an iterative (and thus, computationally expensive) situation. Moreover, it is also safe to assure that Agglomerative is the most suitable methodology when structural information of the objects being matched, such as the case of graph correspondences, since it is based on the BP [1] framework.

In this chapter, an upgrade for the M-CF will be presented. We propose the inclusion of weighting parameters that can be learnt from some information known beforehand to give a specific quality values to the correspondences and the cost functions involved. Taking into consideration that the consensus framework works with two restrictions (loss function and regularisation term), this proposal adds two novel parameters that are not only able to gauge the importance of each restriction, but also to gauge the information provided by each of the involved parties.

This new framework will be onwards referred as the Multiple-Input Consensus Framework through Learning Weights (M-CF+W).

7.2 Definitions and Notations

Suppose two sets of objects $\hat{G} = \{G^1, \dots, G^k, \dots, G^M\}$ and $\hat{G}' = \{G'^1, \dots, G'^k, \dots, G'^M\}$ have been established by separate entities, and several correspondences $f^1, \dots, f^k, \dots, f^M$ have been conformed respectively. Each of the objects is composed of elements $\hat{G}^k = \{v_1^k, v_2^k, \dots, v_{n^k}^k\}$ and $\hat{G}'^k = \{v'^k_1, v'^k_2, \dots, v'^k_{n^k}\}$.

The function proposed in this chapter is once again composed of two restrictions: a loss function $\nabla(f)$ plus a regularisation term $\Omega(f)$. If $\nabla(e) \in \mathbb{R}^N$ and $\Omega(e) \in \mathbb{R}^N$, then the inclusion of two weights α and β gauge how much these restrictions have to be imposed. Taking into consideration that each of the restrictions is composed of multiple parties, the weights also can gauge how these parties affects the whole function overall. Thus, $\alpha \in \mathbb{R}^N$ and $\beta \in \mathbb{R}^N$ are defined as follows

$$\bar{f}^* \quad = argmin_{\forall f} \{\langle \beta, \nabla(f) \rangle + \langle \alpha, \Omega(f) \rangle\} \tag{7.1}$$

Given the definitions used so far for the loss function and the regularisation term, α and β can be defined as a vectors of values, where each value $\alpha^k = \{\alpha^1, \dots, \alpha^k, \dots \alpha^M\}$ gauges the distance to a specific correspondence $dist(f, f^k)$, while $\beta^k = \{\beta^1, \dots, \beta^k, \dots \beta^M\}$ gauges the cost of each correspondence $Cost(G, G', f^k)$. These parameters could be automatically learnt from the database at hand; this means that both the correspondences' distance and their costs must be known in order to perform a training step and learn this values.

7.3 Methodology

First, the process to learn weighting parameters $\alpha = (\alpha^1, \dots, \alpha^k, \dots, \alpha^M)$ and $\beta = (\beta^1, \dots, \beta^k, \dots, \beta^M)$ will be described. These weights represent the level of confidence that exists on the initial correspondence F^k and cost matrices C^k respectively. To begin, it is important to establish that both parameters α^k and β^k have to be high if we believe matrices F^k and C^k are properly defined. That is, that f^k is a correspondence with a high number of correct mappings, and that the cost between features $C^k_{i,j}$ really represents the dissimilarity between v^k_i and v'^k_j elements. Also, notice that the learning algorithm does not force the user to impose the M correspondences simultaneously, and therefore it could be applicable in fields where this data is not available at the same time. This implies that each of the k^{th} elements of α^k and β^k can be updated separately.

The learning set is composed of several registers, each one being a quartet with a structure, $\{G^k, G'^k, f^k, \check{f}^k\}$. As mentioned before, G^k and G'^k are sets of element that have the k^{th} feature, f^k is the proposed correspondence, and $\check{f}^k$ is a ground truth correspondence between these sets. Several registers can have sets of some specific feature k, with weights α^k gauging the quality of the correspondence. These are calculated through a similarity function between the correspondence f^k and the ground truth correspondence $\check{f}^k$, where the similarity is calculated as the inverse of the HD. Notice that the obtained value does not depend on the sets G^k and G'^k, but only on the correspondences.

Meanwhile, weights β^k gauge the quality of the features. They consider the costs between elements $C_{i,j}^k$ and the ground truth $\check{f}^k$, but the computed correspondence f^k is not used. That is, only genuine cells in $C_{i,j}^k$ are considered as the ones such that $\check{f}^k(v_i^k) = v'^k_j$, while impostor cells are taken otherwise, that is $\check{f}^k(v_i^k) \neq v'^k_j$. Then, two histograms $H_{genuine}^k$ and $H_{impostor}^k$ are constructed with these cells. The more distant both histograms are, the better these features are represented on the ground truth correspondence. Therefore, β^k is obtained as the distance between these two histograms. This distance is computed as the Earth Movers' Distance between histograms [2]. We decided to use this distance instead of the typical Mahalanobis' distance between probability density functions because in the first samples, the approximation error was very high.

Algorithm 7.1 computes weights α^k and β^k. The first time it is called, the meta-parameters for all k are: $S^k = 0, M^k = 0, H_{genuine}^k = 0$ and $H_{impostor}^k = 0$.

Algorithm 7. 1. Learning Weights

```
Input: $H_{genuine}^k, H_{impostor}^k, S^k, M^k, G^k, G'^k, f^k, \check{f}^k$
Output: $\alpha^k, \beta^k, H_{genuine}^k, H_{impostor}^k, S^k, M^k$
Begin
$M^k++$
$C = CostMatrix(G^k, G'^k)$
$S^k = S^k + \frac{1}{HammingDistance(f^k, \check{f}^k) + 1}$
$\alpha^k = \frac{S^k}{M^k}$
$H_{genuine}^k = H_{genuine}^k + histogram_{genuine}(C)$
$H_{impostor}^k = H_{impostor}^k + histogram_{impostor}(C)$
$\beta^k = \frac{EarthMoversDistance(H_{genuine}^k, H_{impostor}^k)}{M^k}$
End algorithm
```

Given that Agglomerative has been selected as the most suitable solution, it is proposed to modify its function into:

$$\bar{f}^* = \underset{\forall f}{\operatorname{argmin}} \left\{ \left[-\sum_{k=1}^{M} \alpha^k \cdot F^k + \beta^k \cdot \sum_{k=1}^{M} C^k \right] \circ \mathrm{F} \right\} \tag{7.2}$$

Algorithm 7.2 presents the M-CF+W. It is composed of three main steps. First, the weights α and β to be used are normalised. Then, matrices $F^1, \dots, F^M, \dots, F^M$ and $C^1, \dots, C^k, \dots, C^M$ are computed as explained in the previous chapter. Finally, a LAP solver such as the Hungarian method [3], the Munkres' algorithm [4] or the Jonker-Volgenant solver [5] is applied on the resulting matrix H.

Algorithm 7. 2. M-CF+W

```
Input: ${f^1, \dots, f^k, \dots, f^N}, {G^1, \dots, G^k, \dots, G^N}, \{G'^1, \dots, G'^k, \dots, G'^N\}$
Output: $\bar{f}^*, G, G'$
Begin
$\alpha'^k, \beta'^k = Normalise(\alpha^k, \beta^k)$
for $i, j, k$
    $F^k[i,j] = 1 \;\; if$ if $f^k(a_i^k) = a'^k_j$
    $F^k[i,j] = 0 \;\; otherwise.$
end for
for all $k$
    $C^k = CostMatrix(G^k, G^{k\prime})$
end for
$H = \beta'^k \cdot \sum_{k=1}^{N} C^k - \sum_{k=1}^{N} \alpha'^k \cdot F^k$
$\bar{f}^* = LinearSolver(H)$
End algorithm
```

7.4 Experimental Validation

Once the database used is properly described, we propose two experimental validations. In the first one, graph correspondences from the most reliable portion of the database are used (in terms of inlier mappings density) to validate the functionality of the framework and to observe whether there is a quality increase with respect to the results presented using the 2-CFBP (chapter 5) or not. In the second validation, salient point correspondences are used from the respective portion of the database, in order to verify if the quality has increased with respect to the M-CF (chapter 6). On both cases, the learned weights will be evaluated in advance.

7.4.1 Database Used

-Tarragona Exteriors Graphs 2: This database is composed of 5 sequences $\times$ 100 pairs of images $\times$ 5 salient point and feature extractors $\times$ 4 matching algorithms $= 10'000$ quartets (DB1), plus 7 sequences $\times$ 25 pairs of images $\times$ 5 salient point and feature extractors $\times$ 4 matching algorithms $= 3'500$

quartets (DB2), which consist of two sets of salient points and two correspondences between them, which are split among 20 datasets (5 extractors $\times$ 4 matching algorithms). The first correspondence has been obtained with an existing matching method, and the second one is the ground truth correspondence. To construct this database, we used the 11 image samples from 5 public image databases called “BOAT”, “EASTPARK”, “EASTSOUTH”, “RESIDENCE” and “ENSIMAG” (DB1) [6], as well as 7 public image databases called “BARK”, “BIKES”, “GRAF”, “LEUVEN”, “TREES”, “UBC” and “WALL” (DB2) [7]. The original databases are composed of sequences of images taken of the same object, but from different camera angles and positions. Together with the images, the homography estimations that convert the first image (img00) of the set into the other ones (img01 through img10 in case of DB1 and img01 through img06 in case of DB2) is provided. Notice that from this information we are capable of also deducing the homography between any pair of images. From each of the images, we extracted the 50 most reliable salient points using 5 methodologies: FAST [8], HARRIS [9], MINEIGEN [10] SURF [11] (native Matlab 2013b libraries) and SIFT [12] (own library). A total of 50 correspondences between all combinations of pairs of images within a sequence is computed through the use of four algorithms: Matlab’s *matchFeatures* function [13] (with the parameter $MaxRatio = 1$ to ensure the maximum number of inlier mappings) and the FBP method [14] applying three different node centralities (chapter 2): a) Node ($K_v = 0.2$) b) Degree ($K_v = K_e = 0.2$) and c) Clique ($K_v = K_e = 0.2$). The insertion and deletion costs used for the graph matching methods were set to, in such way that a fair number of inlier mappings is achieved. Finally, we construct the ground truth correspondence using the homography provided by the original image databases. In summary, the database, divided into two sub-databases identified as DB1 and DB2, has a total 20 datasets with a total of $13'500$ quartets (675 quartets per dataset) composed of two sets of features and two correspondences $\{G_i, G_i', f_i, h_i\}$, where $i \in [1, \ldots, 13'500]$.

The “Tarragona Exteriors Graph 2” database is available in [15].

7.4.2 Validation in Graph Correspondences

To make a fair comparison with the results obtained using the 2-CFBP (chapter 5), only DB1 with the registers where img00 is the output graph are used.

As commented before, the first step when applying the new framework is to learn α and β. Notice that due to the database construction, α^{k_α} weights for $1 \leq k_\alpha \leq 20$ shall be learnt for every possible combination of feature extractor $\times$ matching algorithm, while only β^{k_β} for $1 \leq k_\beta \leq 5$ need to be learnt, given that only 5 feature extractors where used. Moreover, to better understand the evolution of the parameter learning as more information is considered, we resort to an iterative learning approach. This means that we select each time the first $1 \leq I \leq 4$ quartets of each dataset, and then the algorithm deduces α^{k_α} and β^{k_β} only from such data. In each iteration, the whole parameters are updated.

To show synthesised results, instead of plotting the 20 learnt parameters α^{k_α} for each possible correspondence, in figure 7.1 we grouped the parameters according to the feature extractor and show the average progression of the value of α^{k_α} as more data is used for learning. An iteration $I = 0$ means that no learning has taken place yet, and thus $\alpha^{k_\alpha} = 1/20$ has been imposed for all weights. There is a clear correlation between the results previously observed in the validation of the individual correspondences (chapters 4 and 5) and the information shown in this plot. For instance, it had been proven that correspondences using the SURF feature extractor obtain better results. Therefore, the learning algorithm automatically tends to assign higher values to $\alpha^4, \alpha^9, \alpha^{14}$ and α^{19}; the weights of correspondences that are associated to SURF.

Figure 7.2 shows the progression of β^{k_β} for the five feature extractors. Once again, the learning algorithm assigns a higher value to β^4; the weight associated with SURF. Furthermore, notice that the learning algorithm has assigned much larger values to β^{k_β} than to α^{k_α}, implying that this factor must be taken more into account for the representative prototype search.

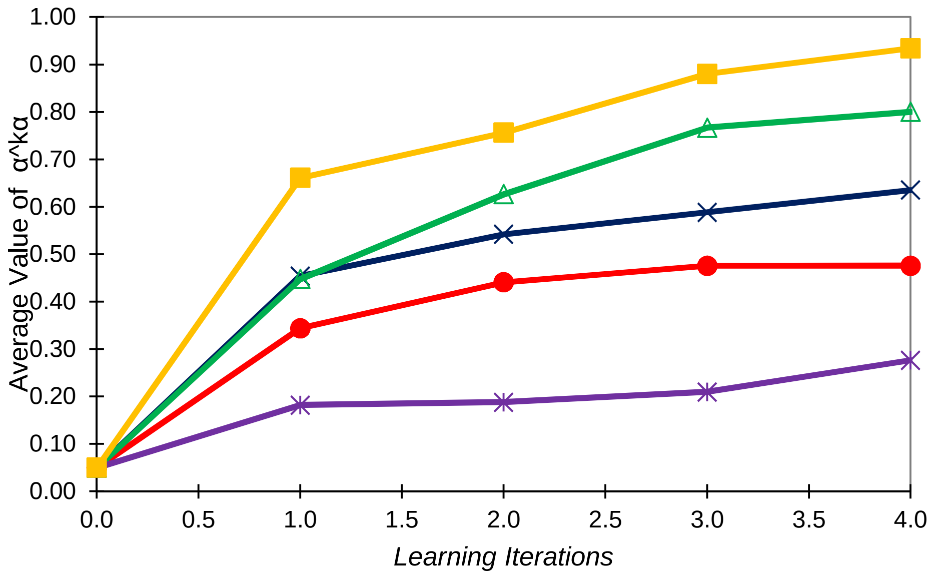


Fig. 7. 1. Evolution of α^{k_α}; (●) FAST-ANY $\overline{\{\alpha^1,\alpha^6,\alpha^{11},\alpha^{16}\}}$, (×) HARRIS-ANY $\overline{\{\alpha^2,\alpha^7,\alpha^{12},\alpha^{17}\}}$, (Δ) MINEIGEN-ANY $\overline{\{\alpha^3,\alpha^8,\alpha^{13},\alpha^{18}\}}$, (■) SURF-ANY $\overline{\{\alpha^4,\alpha^9,\alpha^{14},\alpha^{19}\}}$, ($*$) SIFT-ANY $\overline{\{\alpha^5,\alpha^{10},\alpha^{15},\alpha^{20}\}}$.

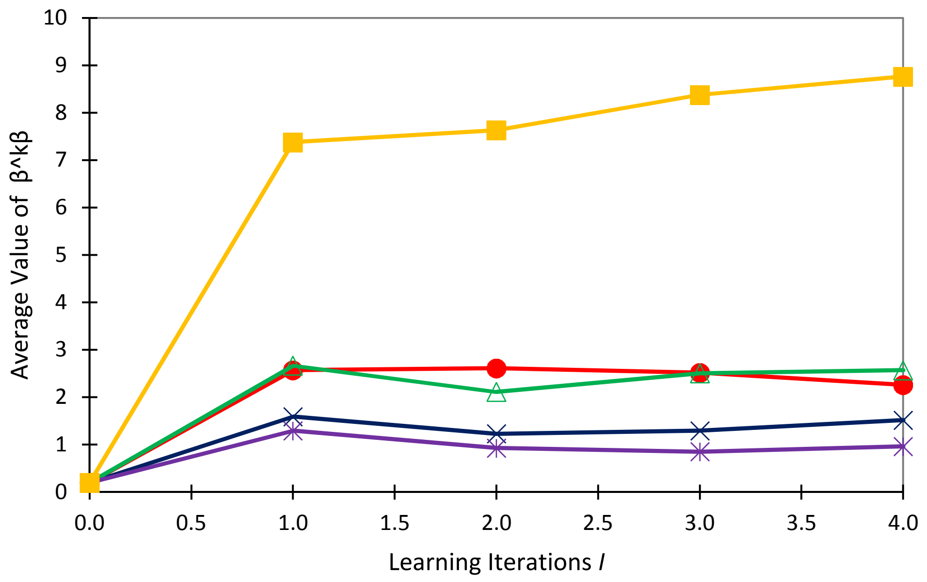


Fig 7. 2. Evolution of β^{k_β}; (●) FAST β^1, (×) HARRIS β^2, (Δ) MINEIGEN β^3, (■) SURF β^4, ($*$) SIFT β^5.

Table 7.1 shows the accuracy performance of the 2-CFBP in terms of correct mappings. $E_{a,b}$ represents the combination of feature extractors a and b, where 1: FAST, 2: HARRIS, 3: MINEIGEN, 4: SURF and 5: SIFT.

Table 7. 1. Average number of correct mappings obtained by the 2-CFBP on the 20 datasets and the combinations of the five feature extractors (in bold the highest values).

Dataset	$\mathbf{E_{1,2}}$	$\mathbf{E_{1,3}}$	$\mathbf{E_{1,4}}$	$\mathbf{E_{1,5}}$	$\mathbf{E_{2,3}}$	$\mathbf{E_{2,4}}$	$\mathbf{E_{2,5}}$	$\mathbf{E_{3,4}}$	$\mathbf{E_{3,5}}$	$\mathbf{E_{4,5}}$	Average
1	84	78	104	70	96	116	89	**128**	57	80	90.2
2	74	72	82	60	68	109	61	**111**	52	42	73.1
3	70	65	84	63	56	**89**	70	62	42	37	63.8
4	10	10	48	9	9	50	9	46	9	**66**	26.6
5	35	27	84	36	22	**94**	43	83	41	85	55
6	29	27	31	27	33	**42**	**42**	35	40	32	33.8
7	11	23	43	35	18	38	32	**46**	37	**46**	32.9
8	2	1	38	4	2	47	1	44	2	**67**	20.8
9	8	19	20	12	16	27	11	27	10	**30**	18
10	11	7	**34**	18	3	29	14	25	12	30	18.3
11	9	2	18	10	4	17	10	**19**	12	18	11.9
12	1	1	22	1	1	20	1	22	1	**32**	10.2
13	32	24	124	27	25	**129**	26	122	23	100	63.2
14	16	19	**36**	26	12	26	20	29	21	21	22.6
15	10	15	**24**	15	12	**24**	16	**24**	19	17	17.6
16	1	1	59	1	2	51	1	39	1	**106**	26.2
17	4	3	47	4	5	44	6	43	5	**52**	21.3
18	4	5	34	3	1	**37**	5	35	3	35	16.2
19	2	3	30	5	3	**37**	6	31	5	36	15.8
20	1	1	42	1	1	38	1	36	1	**53**	17.5

Table 7.2 shows the average number of correct mappings for three different approaches. The first one is the average number of correct mappings obtained by the correspondences individually (as previously shown in table 5.2). The second one is the average of 2-CFBP shown in table 7.1. Finally, the last column shows the average number of correct inliers detected by the M-CF+W. Notice that for the 20 datasets, the number of correct inliers of the new learnt consensus correspondence is higher than the past approaches. In addition, the quality is still kept higher for correspondences which use the CLIQUE local sub-structure. When applying the M-CF+W, it is only required to use β^{k_β} with the respective matching algorithm that includes such feature extractor.

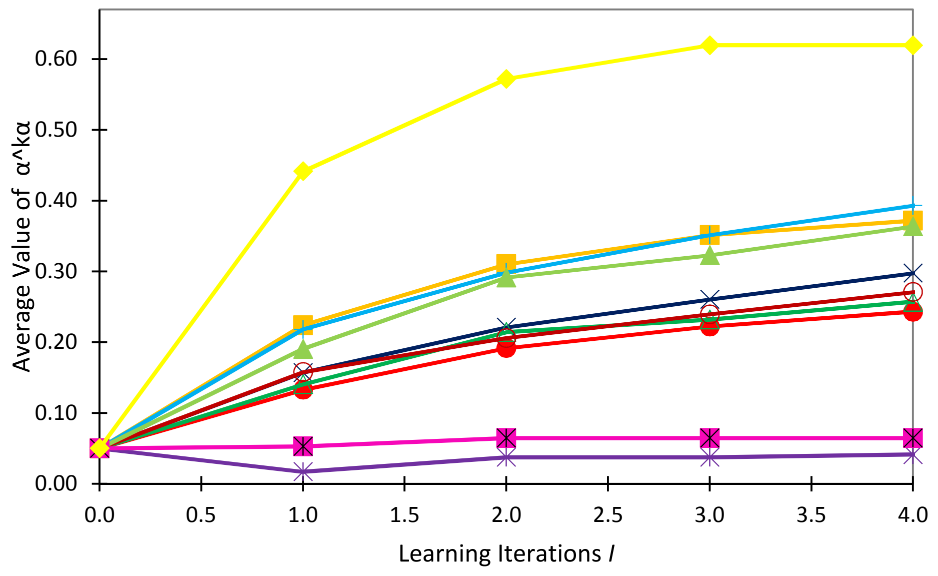
Table 7. 2. Average number of correct mappings comparing the individual correspondences (table 5.2), 2-CFBP, and M-CF+W.

Dataset	Name	Individual	M-CF	M-CF+W
1	BOAT_BP-CLIQUE	49.8	90.2	234
2	BOAT_BP-DEGREE	40.8	73.1	222
3	BOAT_BP-NODE	39.2	63.8	192
4	BOAT_matchFeatures	18	26.6	99
5	EASTPARK_BP-CLIQUE	26.6	55	137
6	EASTPARK_BP-DEGREE	23.4	33.8	117
7	EASTPARK_BP-NODE	19.6	32.9	112
8	EASTPARK_matchFeatures	14.4	20.8	63
9	EASTSOUTH_BP-CLIQUE	8.4	18	32
10	EASTSOUTH_BP-DEGREE	10	18.3	31
11	EASTSOUTH_BP-NODE	7.8	11.9	21
12	EASTSOUTH_matchFeatures	8	10.2	19
13	RESIDENCE_BP-CLIQUE	31.4	63.2	147
14	RESIDENCE_BP-DEGREE	14.8	22.6	122
15	RESIDENCE_BP-NODE	11.8	17.6	107
16	RESIDENCE_matchFeatures	22	26.2	78
17	ENSIMAG_BP-CLIQUE	10.8	21.3	59
18	ENSIMAG_BP-DEGREE	9	16.2	43
19	ENSIMAG_BP-NODE	8.2	15.8	27
20	ENSIMAG_matchFeatures	11.4	17.5	24

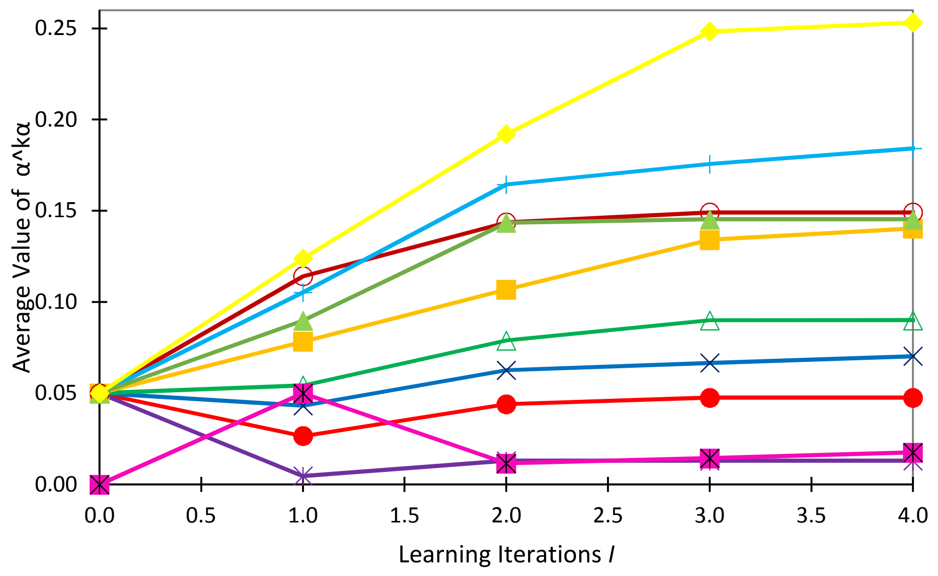
7.4.3. Validation in Salient Point Correspondences

Since the salient point correspondences have a larger inlier mapping density for this specific case, this second validation uses the portion “Tarragona Database Graphs 2” were salient point correspondences are generated, meaning that only instances with mappings generated with *matchFeatures* (MF) or BP-Node (BP) matching algorithms will be used. Moreover, in this validation there will be distinction between matching algorithms, meaning that each consensus experiment involves a total of 10 correspondences and that weighting parameters α^{k_α} will be learnt a total of $1 \leq k_a \leq 10$ times. Still, β^{k_β} is learnt $1 \leq k_a \leq 5$ times since no additional extractors have been included.

Figures 7.3 and 7.4 show the average values of parameters α^{k_α} and β^{k_β}. On both cases, the values of each sub-dataset are shown separately, due to the learning process being done this way. Iteration 0 means that no learning has taken place yet, and the values $\alpha^k = 1/10$ and $\beta^k = 1/10$ have been imposed. In each iteration, the whole parameters are updated. Weights associated to correspondences generated through BP and the SURF feature extractor (α^9), along with SURF weights (β^4) in the case of costs, are set with higher values as iterations increase.

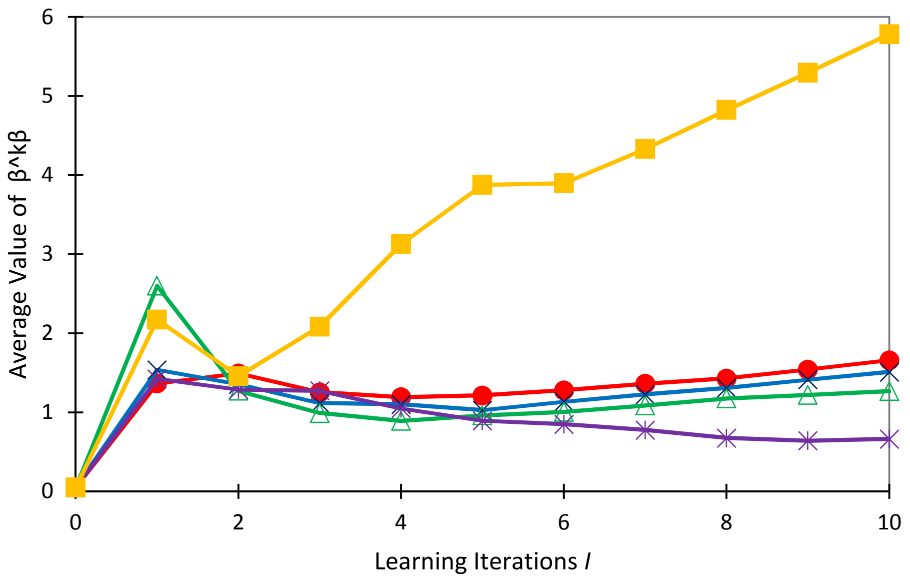


a)

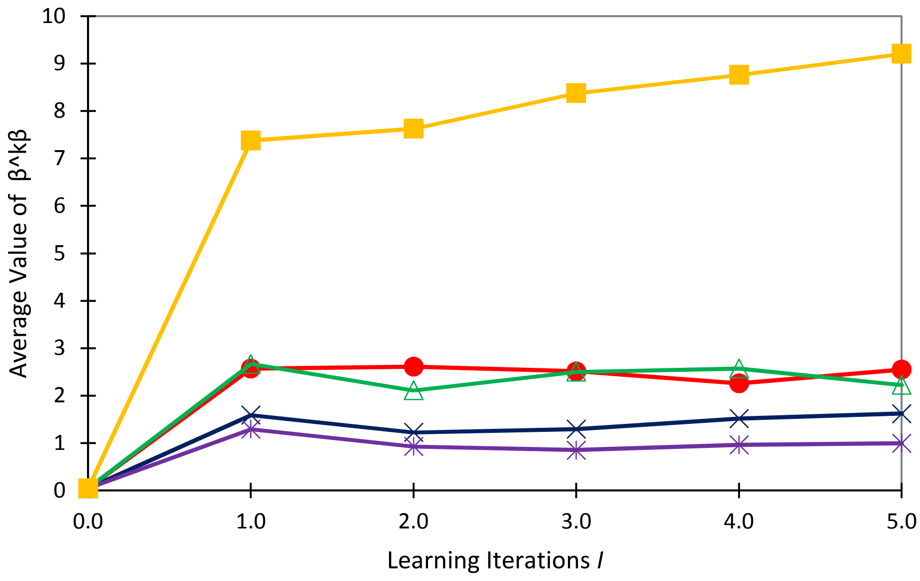


b)

Figure 7.3. Evolution of α^{k_a} in a) DB1 and b) DB2; (●) FAST-MF α^1, (×) HARRIS-MF α^2, (△) MINEIGEN-MF α^3, (■) SURF-MF α^4, (∗) SIFT-MF α^5, (O) FAST-BP α^6, (+) HARRIS- BP α^7, (▲) MINEIGEN- BP α^8, (♦) SURF- BP α^9, (☒) SIFT- BP α^{10}.



a)



b)

Figure 7. 4. Evolution of β^{k_β} in a) database 1 and b) database 2; (●) FAST β^1, (×) HARRIS β^2, (Δ) MINEIGEN β^3, (■) SURF β^4, (∗) SIFT β^5.

To have a frame of reference for the accuracy increase, table 7.3 shows the classification ratio (measured in %) of the 10 combinations given DB1 and DB2 given each configuration to generate a correspondence.

Table 7.3. Recognition ratio in % of each combination (feature × matching algorithm) given the two databases.

DB	FAST MF	HARRIS MF	MINE MF	SIFT MF	SURF MF	FAST BP	HARRIS BP	MINE BP	SIFT BP	SURF BP
1	15	15	14	1	17	17	20	20	2	22
2	5	5	5	1	10	14	13	14	2	15

Table 7.4 shows how these new weights influences the recognition ratio. For each database, we present four results: without learning (the whole weights take the same value), learning only α, learning only β, and learning both weights. These weights are obtained at the fifth iteration of the learning algorithm for both sub-datasets. There is an important increase in the classification rate when both weights are learned with respect to when weights are not learned. Note that learning α makes the ratio increase but this is not the case for weight β. Nevertheless, the accuracy when both weights are learned is higher than when only α is learned.

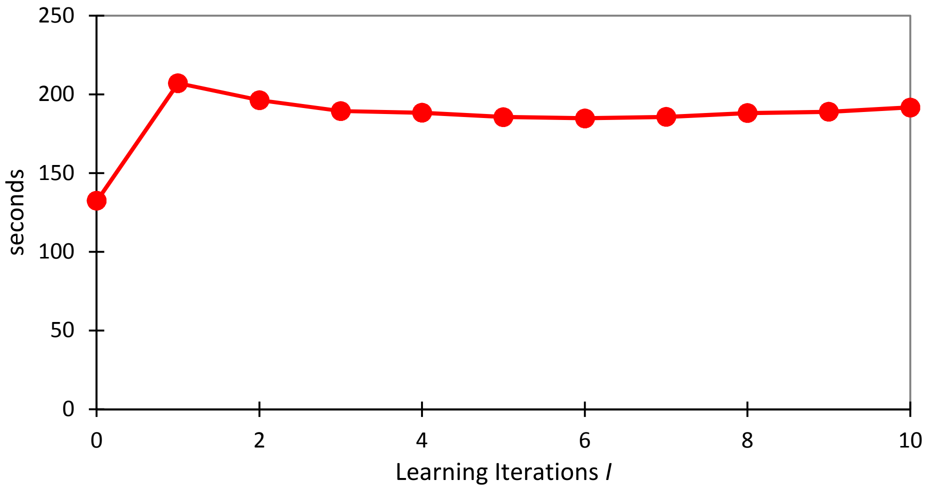
Table 7.4. Correct inlier mappings in %, considering four learning options.

DB	No learning	Learning α	Learning β	Learning α and β
1	25	45	23	53
2	21	25	19	31

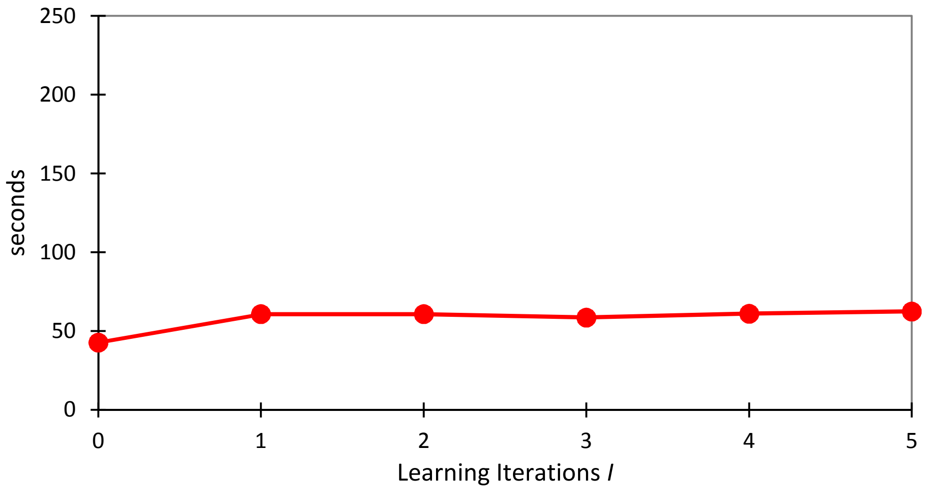
It is important to know whether the addition of the learning weights algorithm has affected in the runtime of the consensus framework. Therefore, we show in figure 7.5 the average runtime (in seconds) for the new method to compute one consensus correspondence given our databases. It considers the time to calculate α^k and β^k according to each iteration, plus the time of computation of the consensus itself.

First, as reported in the last chapter, Agglomerative spends an average of 132.56 seconds to compute a correspondence in DB1. This information is present in the left side plot of figure 7.5. When the learning weights algorithm is introduced, the runtime is increased to 207.82 seconds when one learning iteration is configured. This is a considerable increase, however, the quality increases as well. Nonetheless, it is also important to observe that the time consumption does not increase as the number of iterations increases. In fact, it is interesting to point out that, on average for DB1, the first iteration is the one that delays the most. This happens due to the fact that in some cases, the following iterations obtain same values α^k equal to 0. As a consequence, the time to compute the linear solver that leads to the consensus is reduced. With respect to the runtime of DB2, we also detect an increase in runtime from 42.33 seconds in the Agglomerative to

60.98 when the first learning iteration is performed. Afterwards, the runtime is practically kept stable. Tests were performed using a PC with Intel 3.4 GHz CPU and Windows 7 operating system.



a)



b)

Figure 7.5. Evolution of the average runtime (in seconds) to compute a consensus correspondence in a) DB1 and b) DB2.

7.5 Discussion

In this chapter, we have enhanced the consensus framework by including a learning weights algorithm to deduce how much to trust on each correspondence and each cost function involved in the consensus. This learning methodology is based on the ground truth information provided by the database used. Therefore, it has a very specific scope of applications, but is a first insight towards applying machine learning methodologies to better learn a consensus correspondence.

In the experimental section, it is demonstrated that there is a clear dependency between the quality of the correspondences used and the learned weights. The better the original correspondence is, the higher the weights are. This means that the model has been properly formulated and it is applicable to the current scenario. Moreover, it is shown that the accuracy is much better when the weights have been learned than when keeping them uniform. Finally, it is noticed that when applying the learning weights algorithm, regardless of the iterations used, the runtime is roughly kept constant. This is an important fact, since if the learning weights algorithm is applied in an online form, the consensus framework can continue learning the weights each time that more data is available, without having an impact on the computational cost.

Chapter 8 A New Distance and Weighted Mean Search Strategy for Correspondences

8.1 Introduction

So far in this research work, efforts towards learning a consensus correspondence of multiple correspondences have been explained. We have effectively shown that this framework is adequate for this problem to be overcame, highlighting the advantages and disadvantages of using each solution depending on the correspondence scenario found. As a next step, we address the problem of defining a robust distance between correspondences, as well as a new methodology to find weighted mean correspondences.

As explained throughout this work, learning the consensus of multiple correspondences heavily relies on the distance used. Given that the study and use of correspondences as data structures is relatively new, the only distance defined between correspondences so far has been the Hamming distance (HD). Although this option becomes very convenient for its simplicity, it presents one major flaw: it takes into account only the element-to-element mapping as a whole, but not the attributes and structure of the elements that are being mapped.

Consider the scenario shown in figure 8.1. Three mappings contained in three correspondences f^1, f^2 and f^3 map a single node v_1 of an output attributed graph G to three different nodes v'_1, v'_2 and v'_3 of an input attributed graph G' respectively. Moreover, the attributes and/or structure of v'_1 make this node very similar to v'_2, but make both v'_1 and v'_2 very distinct to v'_3.

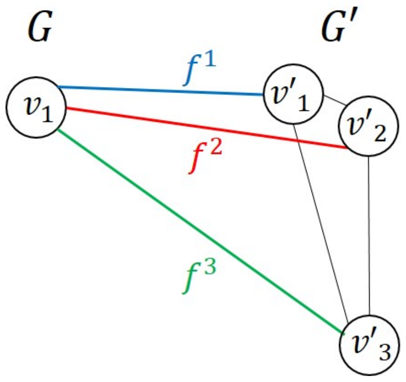


Figure 8. 1. Proposed scenario to explain the problematic of using the HD.

If the HD is calculated between the mappings of these correspondences, this would result in $Dist_{HD}(f^1, f^2) = Dist_{HD}(f^2, f^3) = Dist_{HD}(f^3, f^1) = 1$. Obviously, there is some information being left behind in the HD calculation, since the similarity of the attributes of $v'_1$and v'_2 must somehow reflect in a smaller distance between f^1 and f^2 than the one calculated between these two correspondences and f^3.

As noticed in [1] for the case of the formulation of the ensemble clustering problem, there are two approaches for the distance between correspondences. In the first case, the system only uses information of the correspondences as input, and has no access to the properties of the data structures being mapped. This case is represented by the HD. In the other case, the system would use both the correspondences and the properties of the data structures being mapped (i.e. attributes or local sub-structure in the case of graphs) as the input. This case is represented by a newly proposed distance called Correspondence Edit Distance (CED).

In addition, in this chapter we present a new weighted mean search strategy capable of obtaining a determined number of equidistant weighted means of two correspondences through the definition of either distance. By evaluating the characteristics of the weighted means that are obtained using either HD or CED, it is possible to have an insight regarding which distance is better the correspondence case. As mentioned before, other methods have been presented to compute only one weighted mean in the case of strings [2], graphs [3] and clusters [4]. Moreover, in the present work we have presented an intuitive strategy called *Moving Mappings Uniformly*. However, these methods are based on sequential operations that transform one object into the other. Therefore, these strategies result limited for the case in which we require several equidistant weighted means of two correspondences, such as the Evolutionary Method for the Generalised Median (GM) computation (chapter 3), since many weighted means that are not necessarily in the selected transformation path between the two correspondences could be overlooked. Finally, notice that classical optimisation strategies are not suitable for this search, since this newly proposed strategy must return several equidistant weighted means instead of one.

8.2 Methodology

8.2.1 Correspondence Edit Distance

The CED is a newly proposed distance between correspondences, which in contrast to the HD, considers the attributes of structure and attributes of the elements being mapped. This property makes the distance more appropriate than the HD for the computation of a determined number of weighted mean correspondences, as will be shown later in the experimental section.

Given two objects with structural information (i.e. graphs) $G = \{v_1, \dots, v_N\}$ and $G' = \{v'_1, \dots, v'_N\}$, and two correspondences f^1 and f^2 between them, we want to know how close these correspondences are. Using the new distance proposal, the elements to be considered are not the elements of G and G', but the mappings within the correspondences. More formally, correspondences f^1 and f^2 are defined as sets of mappings $f^1 = \{m_1^1, \dots, m_N^1\}$ and $f^2 = \{m_1^2, \dots, m_N^2\}$, where $m_i^1 = (v_i, v_j')$ and $m_i^2 = (v_i, v_j')$. Notice that even if null elements exist within the objects, they would already be part of G and G' before the correspondence generation occurs and thus, the mappings between them are initially treated like any other mapping.

We establish a set H of bijective functions h between the mappings in f^1 and f^2, where $h(m_i^1) = m_i^2$. Then, in a similar form as in the graph edit distance, the CED would be defined as,

$$Dist_{CED_K}(f^1, f^2) = \min_{h \in H}\{EditCost_K(f^1, f^2, h)\} \tag{8.1}$$

where

$$EditCost_K(f^1, f^2, h) = \sum_{\substack{m_i \in M - \widehat{M} \\ m'_j \in M' - \widehat{M}'}} d(m_i^1, m_j^2) + \sum_{\substack{m_i \in M - \widehat{M} \\ m'_j \in \widehat{M}'}} K + \sum_{\substack{m_i \in \widehat{M} \\ m'_j \in M' - \widehat{M}'}} K \tag{8.2}$$

being M and M' the set of all possible mappings, $\widehat{M}$ and $\widehat{M}'$ being null mappings (mappings that contain null elements on either f^1 and f^2), and d being an application dependent distance function between elements within mappings m_i^1 and m_j^2. If $m_i^1 = (a_k, a_t{}')$ and $m_j^2 = (b_t, b_t{}')$ and in the case that the attribute domain of the elements in G and G' is $\mathbb{R}^{\mathrm{m}}$, d could be defined as $d(m_i^1, m_j^2) = Euclidean(a_k, b_k) + \ Euclidean(a'_t, b'_t)$. In the case of graph correspondences, these properties could be the attributes of the nodes and/or the neighbour edges. Then, the problem of finding the distance d between such attributes could be reformulated as a graph matching problem. To this end, one of the most widely used methods to calculate the distance between two local substructures of two graphs in a suboptimal form is the Bipartite (BP) framework [5].

To better understand this newly proposed distance, let us introduce the following practical example. Figure 8.2 (left) shows an example of two correspondences between two sets of points with 3 elements. Correspondence f^1 (blue) is composed of mappings m_1^1, m_2^1 and m_3^1, and f^2 (red) is composed of m_1^2, m_2^2 and m_3^2. Figure 8.2 (right) shows a set of bijective functions H (green) between f^1 and f^2. After H is defined, the distance is calculated as the sum of distances of all bijective functions h. For this example, it is clear that the cost yielded by h_1 is zero given the two mappings are equal. For the rest of cases, depending on the penalty cost K, it may occur that the system yields the cost of mapping substitutions for the mappings paired in h_2 and h_3, or that the distance results in $4K$ (two mapping deletions and two mapping insertions from f^1 towards f^2).

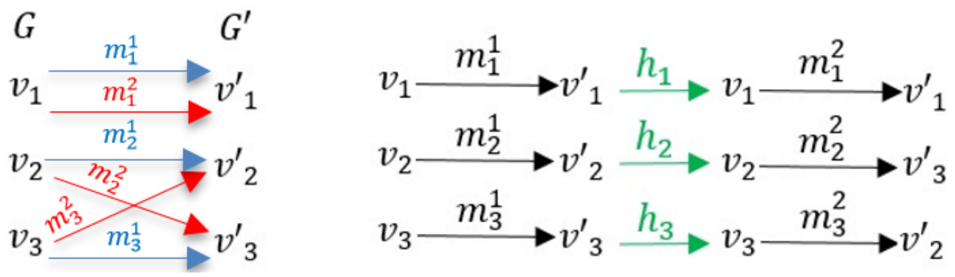


Figure 8. 2. Left: First example of two sets G and G' and two correspondences f^1 and f^2 between them. Right: A bijective function H between f^1 and f^2.

Figure 8.3 shows a second example, this time considering that the third mapping of f^1 selected a null element (ϕ'). As a result, set G was expanded with a null element (ϕ) during the matching process and thus, both correspondences now contain four mappings. For this second case, the cost yielded by the distance function would consist of zero by h_1, plus either a mapping substitution or $2K$ (a deletion and an insertion) by h_2, plus a mapping insertion K by h_3. Finally, h_4 is not considered for the CED calculation since mappings m_4^1 and m_4^2 map a null element of the output set. This restriction is similar to the one stated for the HD in chapter 2.

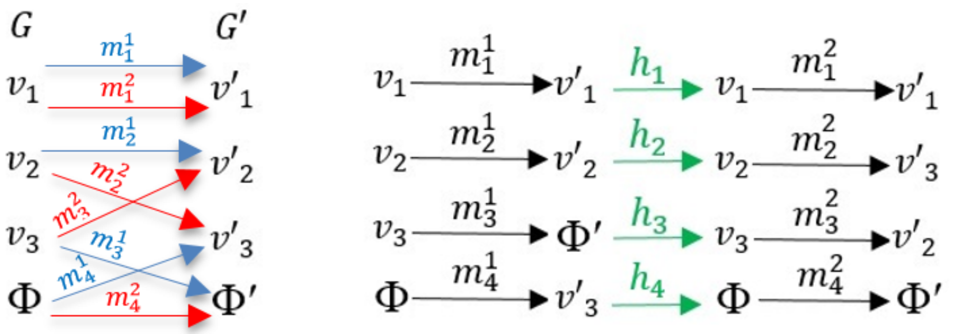


Figure 8.3. Left: Second example of two sets G and G' and two correspondences f^1 and f^2 between them. Right: A bijective function H between f^1 and f^2.

8.2.2 Weighted Mean Search Strategy

As shown in chapter 2, one of the most convenient solutions to find the weighted means between two correspondences is the *Moving Elements Uniformly* strategy, which swaps mappings from f^1 towards f^2 and checks each time whether the conformed combination is a weighted mean or not. This strategy is not convenient if more robust distances such as the CED are intended to be used between correspondences. Furthermore, this search strategy may miss several of the possible weighted means, since it relies on the fact of first finding a weighted mean with a $Dist_{HD} = 2$ (a mapping swap) with respect to f^1, and then sequentially moving towards the second correspondence f^2. Therefore, our goal is to present a new search strategy that regardless of the distance used, given a pair of correspondences f^1 and f^2 between G and G' plus a number Ω, obtains a set of the Ω weighted means $\overline{f^1}, \dots, \overline{f^w}, \dots, \overline{f^\Omega}$ such that $Dist\left(f^1, \overline{f^w}\right)$ is close to α_w. More formally:

$$\begin{aligned} \bar{f}_{\alpha_w} &= \underset{\bar{f} \in T}{argmin}\left\{\left|Dist\left(f^1, \overline{f^w}\right) - \alpha_w\right|\right\} \\ \alpha_w &= Dist(f^1, f^2) \cdot \frac{w-1}{\Omega - 1} : 1 \le w \le \Omega \end{aligned} \tag{8.3}$$

Given two sets of elements of order n, the number of possible bijective correspondences $f \in T$ between them is $n!$. We propose to restrict the problem of searching for weighted means through the whole pool of options in two ways. The first one is related to the distance metric used, and the second one deals with the correspondence exploration space itself. With these restrictions, we aim at reducing the search space to 2^N, making sure that this limitation provides sufficient candidates for the equidistant weighted mean selection.

Firstly, the distance between correspondences $Dist$ must be defined as an addition of local distances

$$Dist(f^1, f^2) = \sum_{i=1}^{N} d(f^1(v_i), f^2(v_i)) \qquad (8.4)$$

where d is a distance measure between local elements. Both correspondence distances considered here hold this restriction. In the case of the HD, the mapping-to-mapping distance is defined such that

$$d(v'_x, v'_y) = \begin{cases} 0 & v'_x = v'_y \\ 1 & v'_x \neq v'_y \end{cases} \qquad (8.5)$$

while in the case of CED it is done in a similar way, but using the structural information of the objects being mapped to obtain the distance values rather than only 0 or 1.

With respect to the second limitation, we define a set of correspondences $\mathcal{W}_{G,G'} \subseteq T_{G,G'}$ (onwards referred as $\mathcal{W}$ instead of $\mathcal{W}_{G,G'}$) and the search of the Ω weighted means is restricted only to this set. Correspondences in this set have as property that their element-to-element mappings are equal to the mappings in f^1 or f^2. That is,

$$\bar{f} \in \mathcal{W} \;\; if \; \bar{f}(v_i) = f^1(v_i) \; or \, \bar{f}(v_i) = f^2(v_i); \;\; \forall \, v_i \in G \qquad (8.6)$$

As a result of considering these two restrictions, the exploration space has been reduced to 2^N. Theorem 1 demonstrates that any correspondence in the new search space $\mathcal{W}$ is indeed a WM of f^1 and f^2.

Theorem 1. If $\bar{f} \in \mathcal{W}$, then the correspondence $\bar{f}$ is a weighted mean of f^1 and f^2.

Proof. Considering the distance definition in equation 8.4

$$\begin{aligned} Dist(f^1, \bar{f}) &= \sum_{i=1}^{N} d(f^1(v_i), \bar{f}(v_i)) \\ Dist(f^2, \bar{f}) &= \sum_{i=1}^{N} d(f^2(v_i), \bar{f}(v_i)) \\ Dist(f^1, f^2) &= \sum_{i=1}^{N} d(f^1(v_i), f^2(v_i)) \end{aligned} \tag{8.7}$$

then, to demonstrate that $\bar{f}$ is a weighted mean correspondence, we have to demonstrate that the weighted mean condition holds,

$$\begin{aligned} &\sum_{i=1}^{N} d(f^1(v_i), f^2(v_i)) \\ &= \sum_{i=1}^{N} d(f^1(v_i), \bar{f}(v_i)) + \sum_{i=1}^{N} d(f^2(v_i), \bar{f}(v_i)) \end{aligned} \tag{8.8}$$

To do so, we demonstrate that next equation holds.

$$\forall_{i,\dots,N}: d(f^1(v_i), f^2(v_i)) = d(f^1(v_i), \bar{f}(v_i)) + d(f^2(v_i), \bar{f}(v_i)) \tag{8.9}$$

If $f^1(v_i) = f^2(v_i)$, then considering equation 8.6 we have $d(f^1(v_i), f^2(v_i)) = d\left(\bar{f}(v_i), f^1(v_i)\right) = d\left(\bar{f}(v_i), f^2(v_i)\right) = 0$ and so the equation holds. Otherwise if $f^1(v_i) \neq f^2(v_i)$, it has to happen that $\bar{f}(v_i) = f^1(v_i)$ or $\bar{f}(v_i) = f^2(v_i)$ due to the restriction of equation 8.8. In the first case, $d\left(\bar{f}(v_i), f^1(v_i)\right) = 0$ and so $d(f^1(v_i), f^2(v_i)) = 0 + d(f(v_i), f^2(v_i))$ and the equation still holds. Finally, the second case, $\bar{f}(v_i) = f^2(v_i)$ is similar to the first one. ■

Algorithm 8.1 returns the best Ω equidistant weighted means $\mathcal{W}_\Omega$ between f^1 and f^2 that are in the set $\mathcal{W}$. The algorithm is composed of two main steps. In the first one, it generates the whole weighted mean correspondences $\mathcal{W}$. In the second one, it selects the best Ω equidistant weighted means from this set, and returns set $\mathcal{W}_\Omega$.

Algorithm 8. 1. Weighted Mean Explore

```
Input: $\Omega, f^1, f^2$
Output: $\mathcal{W}_\Omega$
Begin
        $W = All\_Means(1, partial\_f)$
        $\mathcal{W}_\Omega = Best\_Means(\Omega, \mathcal{W})$
End algorithm

Function $All\_Means(i, partial\_f)$

$N = order\_of\_correspondences$
$if\ i \leq N$
  $if\ f^1(i) \neq f^2(i)$                          $//mappings\ are\ different$
    $for\ j = 1: N$
      $if\ f^1(i) = j \vee f^2(i) = j$               $//search\ for\ j^{th}\ mapping$
        $if\ j\ not\ in\ partial\_f$
          $partial\_f(i) = j$                        $//mapping\ is\ kept$
          $All\_Means(i + 1, partial\_f)$            $//continue\ search$
          $partial\_f(i) = 0$                        $//clear\ i^{th}\ mapping\ in\ partial\_f$
        $end\ if$
      $end\ if$
    $end\ for$
  $else$                                             $//two\ mappings\ are\ equal$
    $partial\_f(i) = f^1(i)$                         $//mapping\ is\ kept$
    $All\_Means(i + 1, partial\_f)$                  $//continue\ search$
  $end\ if$
$else$                                               $//i > N$
    $\mathcal{W} = \mathcal{W} \cup partial\_f$
$end\ if$
End Function.
```

All_Means is a recursive function that after verifying the value of $i \leq N$, compares the mappings of f^1 and f^2. Whenever two i^{th} mappings are equal, then for sure all the possible weighted means must contain such mapping, and this mapping is kept in a partial weighted mean variable called $partial\_f$. Afterwards, the function is executed again for the following mappings. Subsequently, or when the i^{th} mappings are different, the function searches for the mapping of the j^{th} element of G' in either f^1 or f^2. If this mapping has not yet been included in $partial\_f$, then it is stored and the function keeps searching until all combinations of different $partial\_f$ have been computed. These combinations are stored in W.

The second step is composed of function $Best\_Means$. This function sorts the whole weighted mean correspondences obtained in the previous step in ascending order, according to their distance to f^1. Then, the function selects Ω weighted means that are closer to the distance intervals calculated in α_Ω, through the use of equation 8.3.

```
Function $\mathcal{W}_\Omega = Best\_Means(\Omega, \mathcal{W})$

$\mathcal{W}d = empty$
$\alpha_\Omega = empty$
$for\ i = 1{:}\ size(\mathcal{W})$                    //calculate distances of $\mathcal{W}$
        $\mathcal{W}d(i) = Dist(f^1, \mathcal{W}(i))$
$end\ for$
$\mathcal{W}, \mathcal{W}d = sort(\mathcal{W}, \mathcal{W}d)$                 //sort all means and distances
$for\ w = 1{:}\ \Omega$                              //calculate $\alpha_\Omega$ vector
        $\alpha_w = Dist(f^1, f^2) \cdot (w-1)/(\Omega-1)$
$end\ for$
$for\ i = 1{:}\ size(\alpha_\Omega)$          //best means (distance closest to the ones in $\alpha_\Omega$)
        $\mathcal{W}_\Omega(i) = min\{|\mathcal{W}dist - \alpha_w|\}$
$end\ for$
End Function.
```

The worst computational cost of function All_Means is $O(2^N)$. This is because the exploration tree has n levels and each level has, to the most, two branches. Conversely, the cost of function $Best\_Means$ would be equal to $O(N \cdot \log(N))$, since it is a sorting and selecting algorithm.

8.3 Experimental Validation

In this section, three experimental settings are presented. In the first one, we show through an example using graph correspondences generated from images that only by changing from HD to CED directly derives in better distributed weighted means. In a second round of tests, we use synthetic graphs to analyse the average error and runtime that the newly proposed weighted mean search strategy obtains given the two distances. Finally, we observe the influence in the runtime of forcing the weighted means to be within the search space.

8.3.1 Example of the Weighted Mean Quality Improvement using Graph Correspondences

Given the two images on the first register of the "Tarragona Exteriors" database (chapter 4), we select seven salient points and construct two graphs through the Delaunay triangulation as shown in figure 8.4.

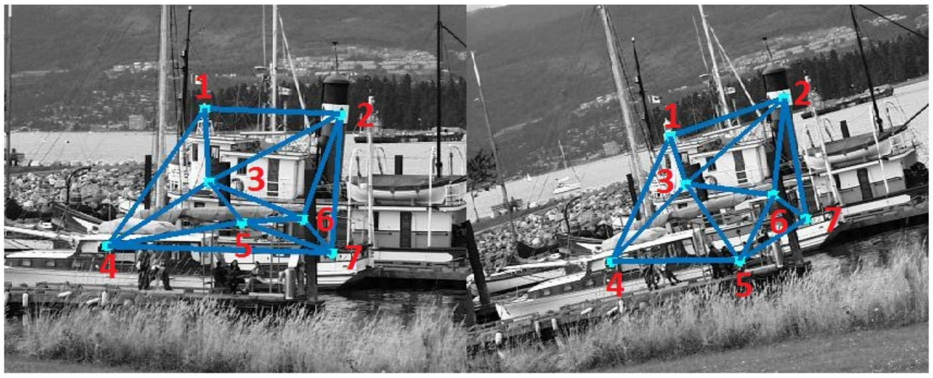


Figure 8. 4. Extracted graphs of the first two images of "Tarragona Exteriors" database.

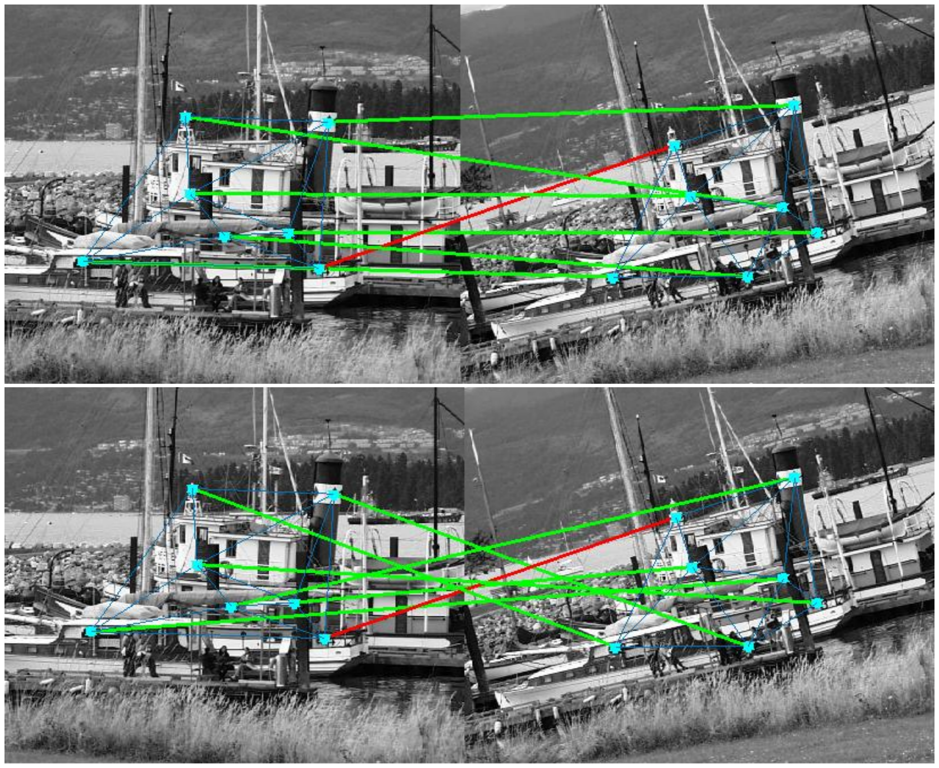
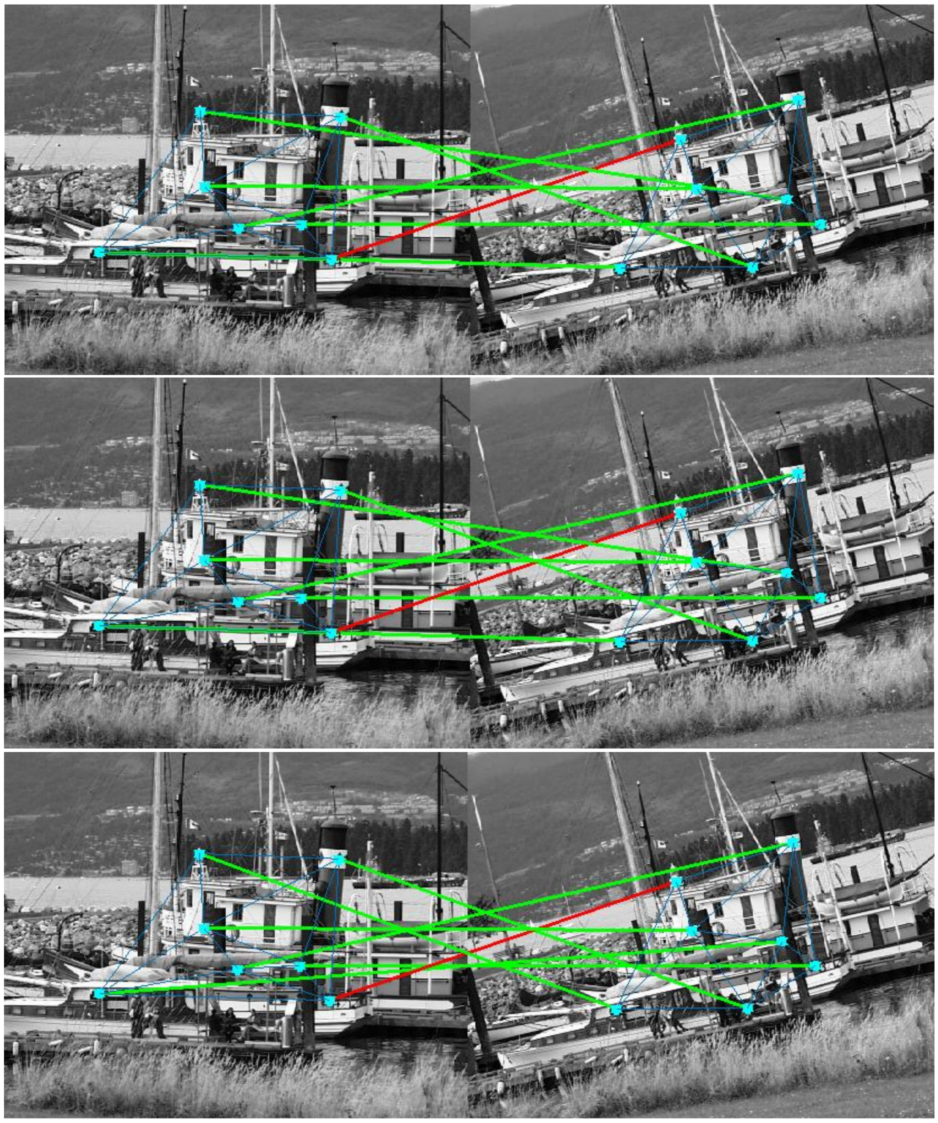


Figure 8. 5. Two correspondences f^1 and f^2 between graphs. The similar mapping in both correspondences is coloured in red.

Afterwards, we generated two correspondences using two different matching algorithms and parameters, which are onwards referred as f^1 and f^2 (figure 8.5). These correspondences have a total of seven node-to-node mappings, with six of them being different one from the other (green lines) and only one being equal (red line).

From these two correspondences, the search strategy has deduced that there are only eight possible weighted means $\mathcal{W} = \overline{f^1}, \dots \overline{f^8}$, from which two of them are the original f^1 and f^2, thus $\overline{f^1} = f^1$ and $\overline{f^8} = f^2$. Figure 8.6 shows the sequence of six correspondences $\overline{f^2}, \dots \overline{f^7}$ that are weighted means in $\mathcal{W}$.



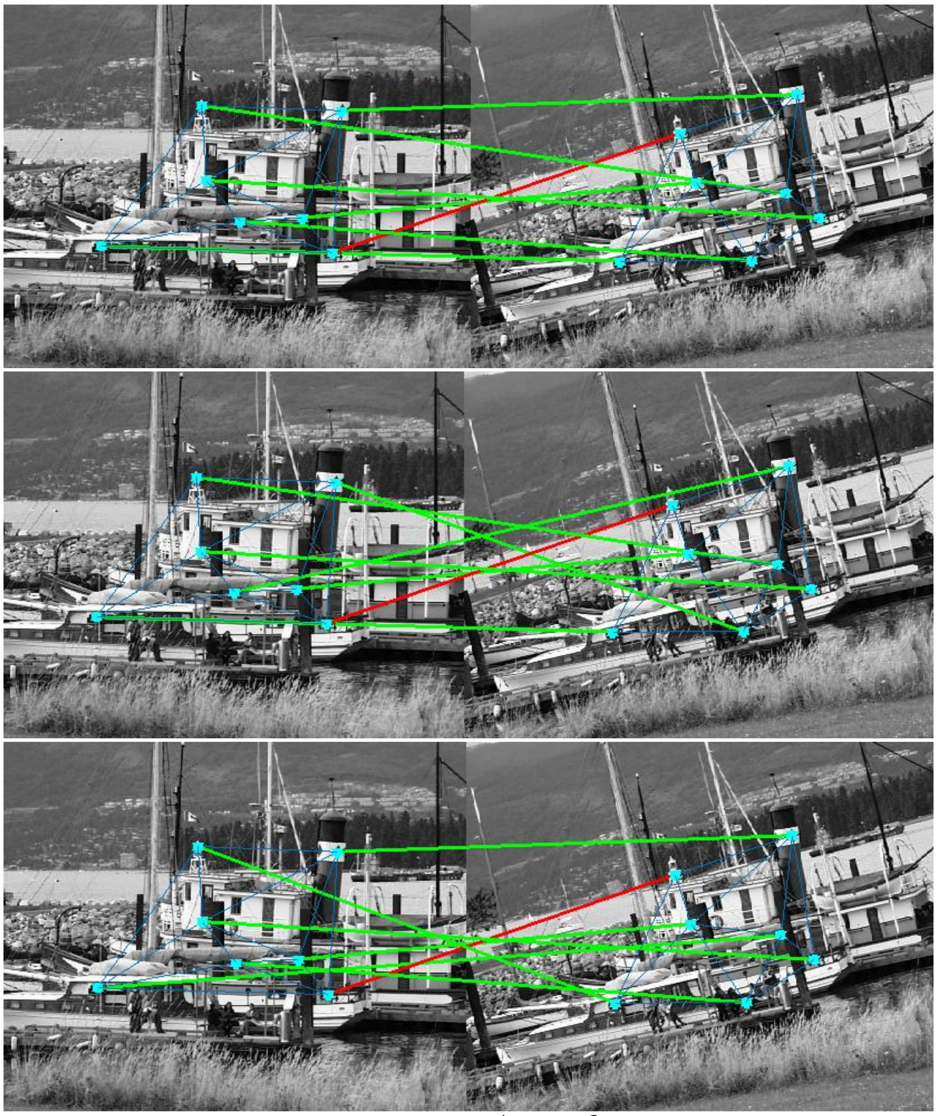


Figure 8. 6. The six weighted means of f^1 and f^2.

Figure 8.7 shows the distance between each of the eight weighted means to f^1, normalised by the distance between f^1 and f^2. That is

$$\alpha = \frac{Dist\left(f^1, \overline{f^w}\right)}{Dist(f^1, f^2)}, 1 \leq w \leq 8 \tag{8.10}$$

where HD and CED have been used separately as the distances. In the case of CED, the node substitution cost is the normalised distance between the FAST features. Moreover, the cost of deleting or inserting a node or an edge is $K = 0.2$ considering the clique local substructure. The total distances between f^1 and f^2 are $HD(f^1, f^2) = 6$ and $CED(f^1, f^2) = 3.668$. As expected, the values obtained by the first and the last weighted mean are zero and one respectively. Note that the HD on the whole set of weighted means only achieves four different values. This is because $HD(f^1,\overline{f^2}) = HD(f^1,\overline{f^3}) = HD(f^1,\overline{f^4}) = 2$, although there is $HD(\overline{f^2},\overline{f^3}) \neq HD(\overline{f^2},\overline{f^4}) \neq HD(\overline{f^3},\overline{f^4})$. The same happens with $\overline{f^5}$, $\overline{f^6}$ and $\overline{f^7}$, but $HD(f^1,\overline{f^5}) = HD(f^1,\overline{f^6}) = HD(f^1,\overline{f^7}) = 4$. The main conclusion obtained from this experiment is that CED is able to deduce more diverse distance values than HD, due to the fact that CED considers the attributes of the graphs while mapping the nodes, but HD does not. In the following lines, we comment on how this property has an effect on the selection of a weighted mean given a specific α.

Suppose that we are interested in the weighted mean closest to $\alpha = 1/2$. Therefore, the selected mean should hold:

$$\bar{f}_{\alpha=0.5} = \underset{\bar{f}_w}{argmin} \left|\frac{Dist(f^1,\overline{f^w})}{Dist(f^1,f^2)} - 0.5\right| \tag{8.11}$$

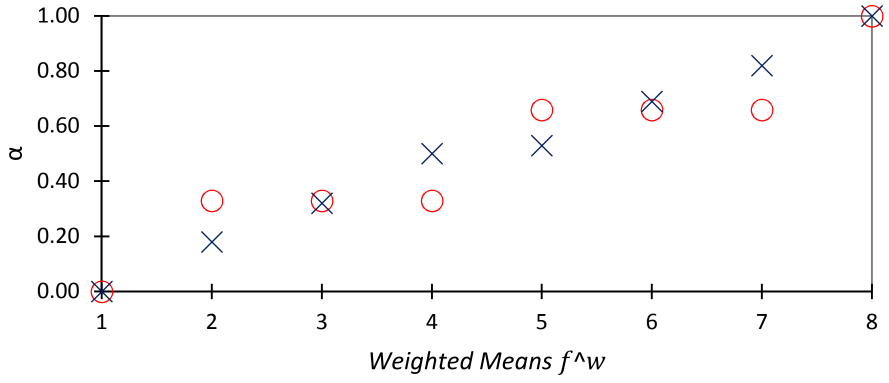


Figure 8. 7. Normalised distances α of the eight weighted means; (O) HD and (×) CED.

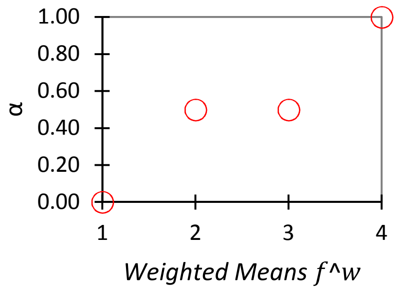
If the correspondence distance used is HD, we deduce from Figure 8.6 that six correspondences minimises $\left|\frac{Dist(f^1,\overline{f^w})}{Dist(f^1,f^2)} - 0.5\right|$ equally, but in the case of CED, only $\overline{f^4}$ minimise this functional. Usually, it is not an issue that the solution is not unique and thus that the algorithm returns indistinctively one of these options. In the case of HD, the error committed by $\overline{f^2}$, $\overline{f^3}$ and $\overline{f^4}$ is $\left|\frac{2}{6} - \frac{1}{2}\right| = \frac{1}{6} = 0.16$, which is the same value than $\left|\frac{4}{6} - \frac{1}{2}\right| = \frac{1}{6} = 0.16$ committed by $\overline{f^5}$, $\overline{f^6}$ and $\overline{f^7}$. Contrarily, for the case of CED, the error committed by $\overline{f^4}$ is $\left|\frac{1.81}{3.668} - \frac{1}{2}\right| = 0.006$.

Table 8.1 shows the mean error when selecting Ω the weighted means using HD or CED. Since the errors of the first and last means are 0, the average error was computed only with the rest of the weighted means. Moreover, we show which weighted means have been selected by both distances. Notice that for all cases, CED errors are lower HD errors. For the cases $\Omega = \{5,6,8\}$, the difference is very small due to the similarity of the selected weighted means by both algorithms. Nevertheless, we conclude error reduction is always achieved by using CED.

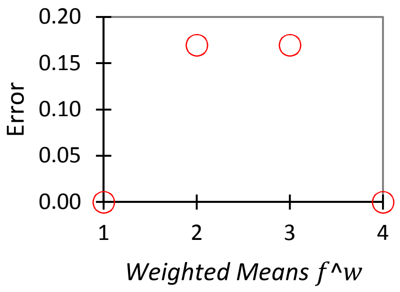
Table 8. 1. Mean error and selected correspondences given HD and CED

Ω	3	4	5	6	7	8
M. Error (HD)	0.166	0.055	0.055	0.055	0.055	0.055
M. Error (CED)	0.006	0.006	0.053	0.052	0.004	0.044
$\mathcal{W}_\Omega$ (HD)	$\bar{f}_1, \bar{f}_5, \bar{f}_8$	$\bar{f}_1, \bar{f}_2, \bar{f}_5, \bar{f}_8$	$\bar{f}_1, \bar{f}_2, \bar{f}_5, \bar{f}_6, \bar{f}_8$	$\bar{f}_1, \bar{f}_2, \bar{f}_3, \bar{f}_5, \bar{f}_6, \bar{f}_8$	$\bar{f}_1, \bar{f}_1, \bar{f}_2, \bar{f}_5, \bar{f}_6, \bar{f}_8, \bar{f}_8$	$\bar{f}_1, \bar{f}_1, \bar{f}_2, \bar{f}_2, \bar{f}_5, \bar{f}_6, \bar{f}_8, \bar{f}_8$
$\mathcal{W}_\Omega$ (CED)	$\bar{f}_1, \bar{f}_4, \bar{f}_8$	$\bar{f}_1 ,\bar{f}_3, \bar{f}_6, \bar{f}_8$	$\bar{f}_1, \bar{f}_3, \bar{f}_4, \bar{f}_6, \bar{f}_8$	$\bar{f}_1, \bar{f}_2, \bar{f}_3, \bar{f}_6, \bar{f}_7, \bar{f}_8$	$\bar{f}_1, \bar{f}_2, \bar{f}_3, \bar{f}_4, \bar{f}_6, \bar{f}_7, \bar{f}_8$	*All*

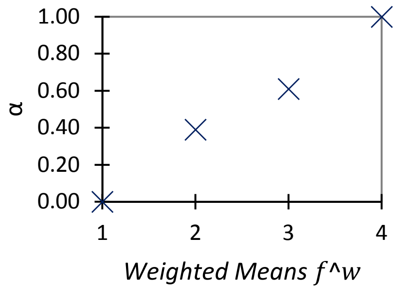
In figure 8.8, in a similar way as figure 8.7, we plot the values of α and the error for each of the selected means, concretising the case when $\Omega = 4$, to confirm that CED errors are lower than HD errors.



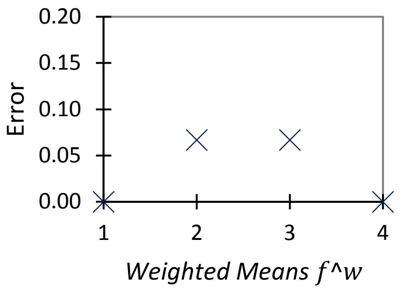
a)



b)



c)



d)

Figure 8. 8. a) HD Alpha, b) HD Error, c) CED Alpha and d) CED Error for the selected means ($\Omega = 4$); (O) HD and (×) CED.

8.3.2 Validation of the Weighted Means using Synthetic Graphs

In this second round of tests, the average error and the runtime obtained using both distances are analysed using the two distances separately. To do so, we randomly generated four sets. Each set is composed of 60 tuples using 6 graph sizes $\times$ 10 graphs per size. Each tuple is composed of a pair of graphs and a pair of correspondences between them. The particularity of each set is the order of the graphs contained. The number of nodes for each set is $N = 5,6, \dots 10$ nodes for the first one, $N = 10,11, \dots 15$ nodes for the second one, $N = 15,16, \dots 20$ nodes for the third one and $N = 20,21, \dots 25$ nodes for the last one. Therefore, the dataset is composed of $4 \times 60 = 240$ pairs of graphs and pairs of correspondences.

The experiments were carried out as follows. Per each tuple, a random α_r was generated such that $0 < \alpha < 1$. After computing all weighted means, the α value of each WM is calculated using HD or CED. Then, the weighted mean with the closest value of α to α_r was selected. Finally, the error was reported. Figure 8.9 shows the relation between the error committed by HD and CED. We realise that for the majority of simulations, the CED error is lower than the HD error. Moreover, this tendency increases when the graph order increases.

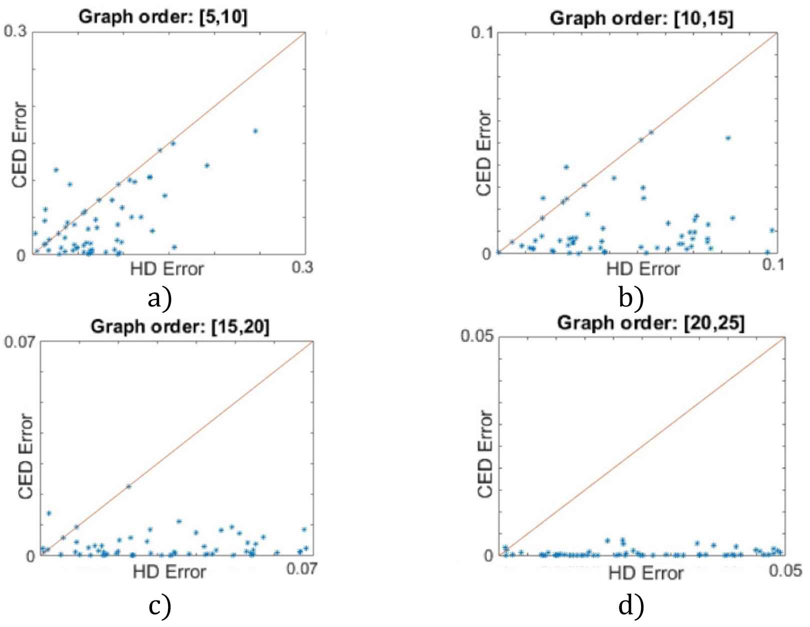
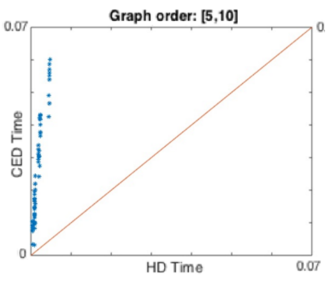


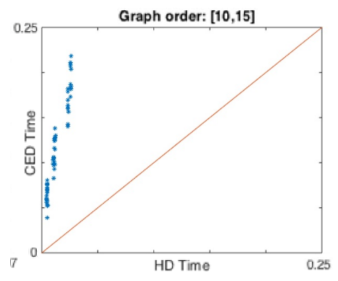
Figure 8. 9. Error committed by HD (x-axis) with respect to CED (y-axis) using a) Graph Order [5,10], b) Graph Order [10,15], c) Graph Order [15,20] and d) Graph Order [20,25]. Note scales are different.

Figure 8.10 shows the relation between the runtime spent to compute the weighted means using the new proposal when either the HD or the CED were used. Runtime was measured in seconds using a PC with Intel 3.4 GHz CPU and Windows 7 operating system. Notice HD is faster in all experimented cases, and this difference is particularly significant in low and medium size graphs. Nevertheless, there is a tendency of reducing the gap between both distances as the order increases. This is because as the order of the graph increases, the runtime of All_Means, which has a cost of $O(2^n)$, becomes larger than the runtime of $Best\_Means$, which has a cost of $O(n \cdot \log(n))$.

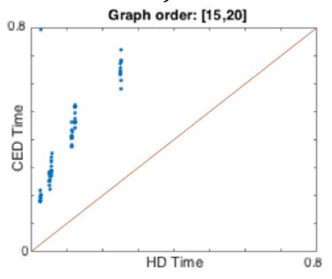
From this experimentation, it is possible to state that in applications with a low number of nodes, HD is a better option since the runtime is clearly faster than CED, considering the error gap is not so important. Furthermore, when the number of weighted means to select is large or the weighted means are required with a high precision, CED becomes once again the better option.



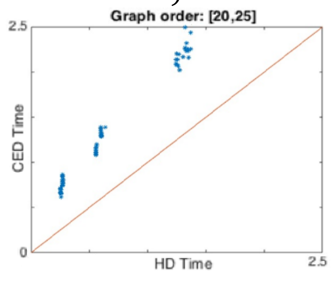
a)



b)



c)

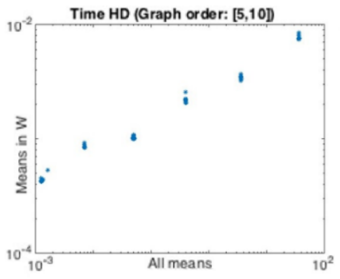


d)

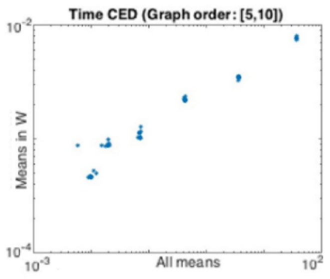
Figure 8. 10. Runtime of the new weighted mean search strategy while deducing the weighted mean using HD (x-axis) with respect to CED (y-axis) using a) Graph Order [5,10], b) Graph Order [10,15], c) Graph Order [15,20] and d) Graph Order [20,25]. Note scales are different.

8.3.3 Validation of the Weighted Mean Search Strategy with respect to the Search Space

The aim of this validation is to analyse the influence on the runtime of forcing the weighted mean search to be within $\mathcal{W}$. We compare the error generated between the new weighted mean search strategy using both distances and an algorithm that generates the whole combinations of possible bijective correspondences given two correspondences. Then, we test if the correspondences obtained by this second algorithm were weighted means. Since the runtime of this second algorithm is higher, experiments were only performed for the first set of graphs of the previous validation. We realised that regardless the distance, both algorithms committed exactly the same error, although in some cases, the returned mean was different since the solution was not unique. Figure 8.11 shows the runtime using both distances in logarithmic scale.



a)



b)

Figure 8. 11. Runtime of generating all weighted means (x-axis) with respect to the new weighted mean search strategy (y-axis) for Graph Order [5,10] using a) HD and b) CED.

8.4 Discussion

It has been emphasised that the definition of a robust distance metric and a weighted mean search strategy are essential not only to correctly learn the consensus of correspondences, but also for the implementation of other uses (i.e. machine learning, kernels, ensemble clustering, clustering embedding, classification and recommender systems). So far in our research, we relied on simple solutions for these two problematics; namely the HD and the *Moving Mappings Uniformly* search strategy. Nevertheless, these options result limited for two main reasons: 1) The HD does not reflect the true dissimilarity of two mapping between elements with different attributes or local substructures and 2) the *Moving Mappings Uniformly* search strategy is not capable of finding either all existing weighted means or an equidistant subset of these, an aspect which could be applied on methods for the GM approximation such as the Evolutionary Method (chapter 3).

In this chapter, we have presented a new distance between correspondences and a new weighted mean search strategy to find a determined number of equidistant weighted means of two correspondences. In the experimental section, both contributions have been validated simultaneously through graph correspondences. We show that when finding a determined number of equidistant weighted means with the new algorithm, the error committed with CED is clearly smaller than the one committed with HD. Conversely, the use of HD results in a faster search than when CED is applied. Nonetheless, this tendency is reduced when the order of the mapped graphs increases. Finally, we have analysed how different the new weighted mean search strategy is with respect to using an algorithm that generates the complete set of weighted mean correspondences. We have noticed that although the error is equal regardless of the used distance, the new search strategy is clearly faster than computing all weighted means.

Chapter 9 Partial to Full Correspondence Generation for Image Registration

9.1 Introduction

Certain areas in computer vision are interested in determining which parts of one image correspond to which parts of another image instead of searching for a whole correspondence. This problem often arises in early stages of applications such as scene reconstruction, object recognition and tracking, pose recovery and image retrieval. In this work, this specific type of image registration problems is defined as partial to full (PF) image registration, and the aim of this chapter is to propose a framework to solve this issue.

One of the most frequent uses of PF image registration is found on biometrics for forensic applications, most notably in palmprint matching. As commented in chapter 2, since the palm contains more features than the fingerprint, identification of subjects is more feasible. Nevertheless, on crime scenes it is more likely to find a small portion of the sample rather than the full palm. For these cases, PF image registration comes as a viable solution for this problem. Another current use of partial to full image registration are the applications that locates elements in outdoor scenes. By using a small part of an image, for instance, a picture taken from a cell phone or an image obtained from social media, manual or automatic systems may be able to find the location of the piece given a larger image, for instance, satellite or surveillance camera images. It is of basic importance to develop effective methods that are both robust in two aspects: being able to deal with noisy measurements and having a wide field of application.

Previous work has been developed exclusively for PF palmprint identification in high-resolution images. These methods differ from the rest mainly because of the methods to pre-process the palmprint images and the features that can be extracted from them. A first approach of PF palmprint matching was presented by Jain & Demirkus in [1]. The method consists of three major components: Region of interest detection, which is only applied to the full palmprint image, feature extraction applied to both images, and feature matching applied to both sets of features. Since it is generally known that the partial palmprints come from specific regions of the palm (i.e. thenar, hypothenar and interdigital), these regions are automatically detected and features from these regions are utilized in the matching phase. The feature extraction phase obtains both the SIFT [2] features of the image and the minutiae. In the feature matching phase, minutiae and SIFT matchers are used in parallel to obtain two different match scores; the score-based fusion is utilized to obtain the final match score. They reported an accuracy of 96% when performing a weighted fusion of minutiae matching and SIFT matching for synthetic (artificially created) partial palmprints of 500×500 pixels. When only using the minutiae features, the recognition rate was around 82%. Acknowledging the fact that SIFT features could not be extracted from partial palmprints printed in surfaces (latent), one year later Jain & Feng [3] presented a method exclusively for latent PF palmprint matching based in minutiae points and orientation field. The alignment is rigid and based on most similar minutiae pairs. Since latent palmprints are more difficult to match than synthetic palmprints, the accuracy decreased to 78.7% using larger partial palmprints (512×877 pixels in average). Also, some additional problems of this approach are the large computational demand on Discrete Fourier Transform and Gabor Filtering, and a requirement of about 150 minutiae per partial palmprint. Therefore, to achieve an acceptable classification rate, authors suggested a fusion of multiple partial palmprints from the same palm. This is an important demand, since sometimes, only one sample is available. In [4], Dai & Zhou presented a method based in minutiae points, ridge density, map, principal map and orientation field. The alignment is rigid and the matching is made through a Hough Transform. Even though they achieve a recognition rate of 91,7% using synthetic partial palmprints of variable size, it is slow in computation. One year later, Dai et. al [5] presented a method which uses the average orientation field for coarse full palmprint alignment and the Generalised Hough Transform [6], [7] for fine segment level alignment, although it needs a manual alignment for partial palmprints. It has an accuracy of 91,9% with a slight improvement of the computational speed.

The framework presented in this chapter, onwards called *PF-Registration*, has drawn inspiration from the aforementioned palmprint literature, but is designed for a general image purpose. This is due to various applications require PF image registration aside from forensics. We believe that this way, we are able take the best aspects from the PF palmprint matching methods and apply them to any kind of image.

9.2 Definitions and Notations

Consider a scenario where the aim is to align an image P that shows a part of another image F. Both images are represented by their salient point locations $(x^P, y^P) = \{(x_1^P, y_1^P), \dots, (x_{n^P}^P, y_{n^P}^P)\}$ and $(x^F, y^F) =$

$\{(x_1^F, y_1^F), \ldots, (x_{n^F}^F, y_{n^F}^F)\}$ together with some other features extracted from these the salient points $v^P = \{v_1^P, \ldots, v_{n^P}^P\}$ and $v^F = \{v_1^F, \ldots, v_{n^F}^F\}$. The number of salient points is n^P and n^F respectively. Due to the nature of the scenario, $n^P \neq n^F$.

The method we propose is based on two main steps. In the first step, K salient points $\{(x_1^c, y_1^c), \ldots, (x_K^c, y_K^c)\}$ on the full image F are selected as candidates to be the centre of the partial image P. Then, the full image F is split in sub-images $F_1, \ldots, F_K$, in which the centre of each image F_a is the candidate position (x_a^c, y_a^c). Each split image F_a is represented by their set of salient point locations $(x^{F_a}, y^{F_a}) = \{(x_1^{F_a}, y_1^{F_a}), \ldots, (x_{n^{F_a}}^{F_a}, y_{n^{F_a}}^{F_a})\}$ and also their corresponding set of features $\gamma^{F_a} = \{\gamma_1^{F_a}, \ldots, \gamma_{n^{F_a}}^{F_a}\}$. Consider that $n^F \leq \sum_{a=1}^{K} n^{F_a}$, since the split images may overlap depending on the selected centres and on the size of the partial image.

In the second step, the algorithm seeks for the best alignment between the salient points (x^P, y^P) of the partial image P and the salient points (x^{F_a}, y^{F_a}) of each of the split images $F_1, \ldots, F_K$. To obtain these alignments, not only the salient point positions are used, but also their extracted features, more precisely, γ^P and γ^{F_a}. Thus, K distances $D_1, \ldots, D_K$, together with K correspondences $f_1, \ldots, f_K$ and K homographies $H_1, \ldots, H_K$ are computed. At the end of this second step, the method selects the image that obtains the minimum distance $D_{P,F}$, and then returns the alignment $H_{P,F}$ and the correspondence $f_{P,F}$ between P and F that obtains this distance. On the following subsections we will explain how each step works. In figure 9.1, we provide a scheme of the framework.

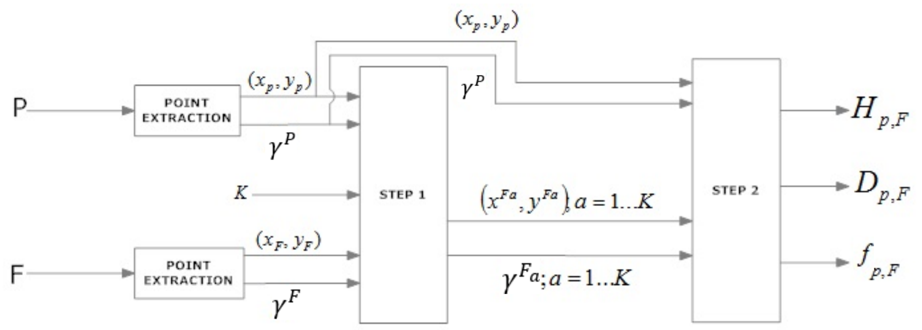


Figure 9. 1. Diagram of the *PF-Registration* framework.

9.3 Methodology

9.3.1 Selecting Position Candidates

Figure 9.2 shows the main structure of the first step of our method. As an input of this step, we use the n^P and n^F salient points (positions and features) of both images that have been previously extracted.

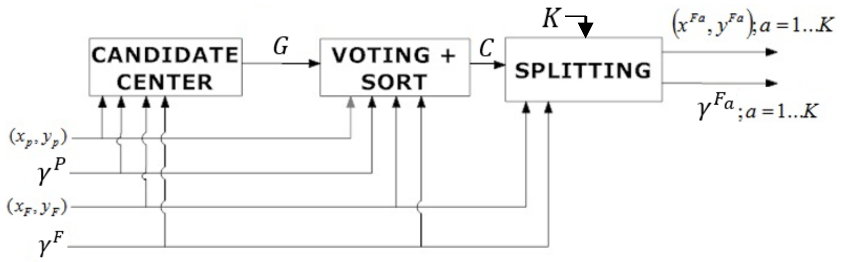


Figure 9. 2. Diagram of step 1 of the *PF-Registration* framework.

The aim of the *Candidate Centre* module is to fill the $n^P \times n^F$ matrix G. Each cell of this matrix represents the position $\left(x_{i,j}^C, y_{i,j}^C\right)$ on F that the centre $(\bar{x}, \bar{y})$ of P would obtain if the point $\left(x_i^P, y_i^P\right)$ was mapped to point $\left(x_j^F, y_j^F\right)$ on F. The most effective form to do it is to apply the Generalised Hough Transform [8], [9]. The method can be configured to use only one or several points to find the centres, and also some information extracted from the features, such as angle information. In the case that an angular feature is available for salient points, then the method is rotation invariant. The aim of this module is to detect the spatial relations on both images. If s points in P and s points in F have the same relative position, then s cells of G will deliver the same candidate centre value.

When matrix G is filled, then the *Voting and Sorting* module generates an ordered list C of the positions $\left(x_{i,j}^C, y_{i,j}^C\right)$ found in G, where $C = \left\{(x_1^c, y_1^c), \dots, (x_{n^c}^c, y_{n^c}^c)\right\}$, using a clustering and voting process. List C is set in a descendent order (centres with the most votes are set first), and n^C is the total number of different centres that have been computed in G. The voting process not only counts the number of centres, but also checks whether the features of two or more candidate centres are similar enough to consider them the same. This process reduces considerably the size of C. To do so, an additional clustering process is performed, which checks whether two centre points $\left(x_{i,j}^C, y_{i,j}^C\right)$ and $\left(x_{i',j'}^C, y_{i',j'}^C\right)$ have to be considered the same given certain closeness threshold. That is, if their spatial distance is lower than a spatial threshold T_s, and if their feature distance is closer than a feature threshold T_v. As a result, this module first counts and orders the cells in G such that $dist\left(\left(x_{ij}^C, y_{ij}^C\right), \left(x_{i'j'}^C, y_{i'j'}^C\right)\right) < T_s$ and $dist\left(f_i^F, f_j^P\right) < T_f$. Both distances, which application dependent, are recommended to have been previously normalised in order to be independent on the scale, rotation and global feature distortions.

Finally, the *Splitting* module selects from C the first K instances, which are considered as the best candidates to be the centre of P on F. Parameter K is application dependent and implies a trade-off between runtime and accuracy. Then, the module splits the salient point location and features of F into K point sets (x^{F_a}, y^{F_a}), $1 \leq a \leq K$ and K feature sets f^{F_a}, $1 \leq a \leq K$. A salient point location $\left(x_k^F, y_k^F\right)$ is included in the set F_a (together with its feature set) if $dist\left(\left(x_k^F, y_k^F\right), \left(x_a^C, y_a^C\right)\right) \leq T_r$, where T_r represents the threshold of the maximum radius of the set. That is, the maximum distance between any point and the centre of the set to be formed. Usually, this threshold is determined depending on the radius of P.

9.3.2 Best Candidate Selection

Figure 9.3 shows the second step of the framework.

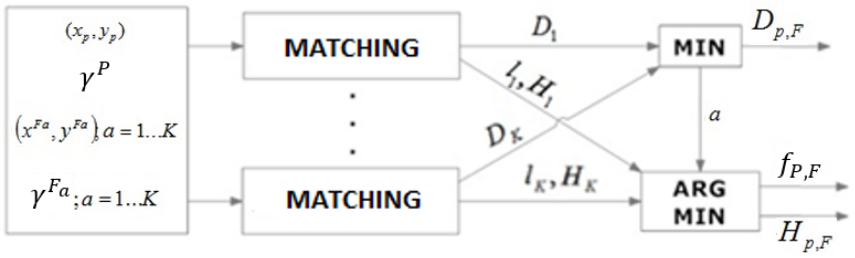


Figure 9. 3. Diagram of step 2 of the *PF-Registration* framework.

In this second step, the framework first seeks in the *Matching* module for the distances $D_a = dist(P, F_a)$; $1 \leq a \leq K$, the correspondences f_a between the points in each set, and also the homographies H_a that transform P to a certain F_a. As commented in chapter 2, several algorithms can be used to find these correspondences [8], [9] and homographies [10], [11]. These algorithms must use the positional information (x^{F_a}, y^{F_a}) and (x^P, y^P) and also their features v^{F_a} and v^P.

Afterwards, in the *Min and ArgMin* module we select the F_a that obtains the lowest distance value D_a. This can be done because D_a is a good enough approximation of $dist(P,F)$. Moreover, given a minimum cost matching approach, the respective correspondence f_a and homography H_a are valid between P and F if the distance has been minimised. Therefore, $f_{P,F} = f_a$ and $H_{P,F} = H_a$.

To conclude, breaking down F into a set of candidates instead of performing straightforward comparison has two important advantages. On the one hand, the computational cost of obtaining the K distances D_a is lower than obtaining directly the value $D_{P,F}$. On the other hand, our method obtains a more precise local minimum, since we only use a certain amount of salient points for each match. In the next section, we discuss the aforementioned statements.

9.4 Reasoning about Complexity and Speedup

Since the algorithms used in this framework are well-known and have been validated, the actual optimality to be discussed in this framework is related to its computational complexity.

The first step of the method is the *Candidate Centre* module, in which the framework seeks for K candidate positions where P is found over P. The complexity cost of this module is $O(n^P \cdot n^F)$, since the calculation of matrix G depends solely on the number of salient points extracted in P and F. Afterwards, the *Voting and Sorting* module generates a list of the centres according to the voting frequency. Once again, an $O(n^P \cdot n^F)$ complexity is observed since this process requires to find the best K candidates within matrix G. At the end of step 1 we encounter the *Splitting* module, where K partial images $F_{1,\ldots,a,\ldots,K}$ are created from the full one. It can be said that the complexity of this module is $O(K \cdot n^P)$, since this process is executed K times, and each F_a has on average n^P salient points.

In the second step, the *Matching* module together with the *Min and ArgMin* module, are found. The first one involves the use of a matching method to find the correspondence, the homography and the distance between the tiny image and the K partial images derived from F. The module's computational complexity is dependent on the selected methods. For example, the complexity of using a matching algorithm based on the Hungarian method [12] would be $O(K \cdot (n^P)^3)$. Conversely, the *Min and ArgMin* module has a constant computational complexity, since finding the minimum distance D_a and its corresponding f_a and H_a can be selected in parallel while the previous module is being executed.

In summary, the highest computational complexity of the *PF-Registration* framework depends on the *Matching* module. In contrast, the computational complexity of a classical registration depends on the matching method, thus $O((n^F)^3)$ if again, the Hungarian method is used. Since it is usual to have a small value of K (for instance $K \leq 4$) and considering that in a partial to full scenario $n^P \ll n^F$, then it can be seen that $O(K \cdot (n^P)^3) \ll O((n^F)^3)$. This inequality shows that the *PF-Registration* framework leads to an important speedup gain with respect to a general image registration method, while maintaining the same characteristics of a classical matching framework.

9.5 Experimental Validation

Once the databases used are properly described, we present two validations. In the first one, we test different matching algorithms to be used in the *Matching* module using a palmprint database. In the second one, a sequence of outdoor scene images is used to validate the capability of applying the framework to a recognition and classification scenario.

9.5.1 Databases Used

-Tarragona Palmprint PF: From the images contained in the "Tarragona Palmprint" database (chapter 4), we split all images into two sets, with the first four palmprints of each subject assigned to the reference set, and the last four to the test set. The palmprint images on the reference set are considered the full image, and the ones on the test set are used to generate circular patches. More formally, these partial palmprints have been created given a variable radius from 100 to 600 pixels and a random minutia as the centre of the patch (the original images have a size of 2040×2040 pixels). To summarise, there are 40 full images on the reference set and other 40 in the test set. From the latter set, 6 circular patches of radius $r = 100$, $200, \dots, 600$ pixels produce a total of 280 partial palmprints.

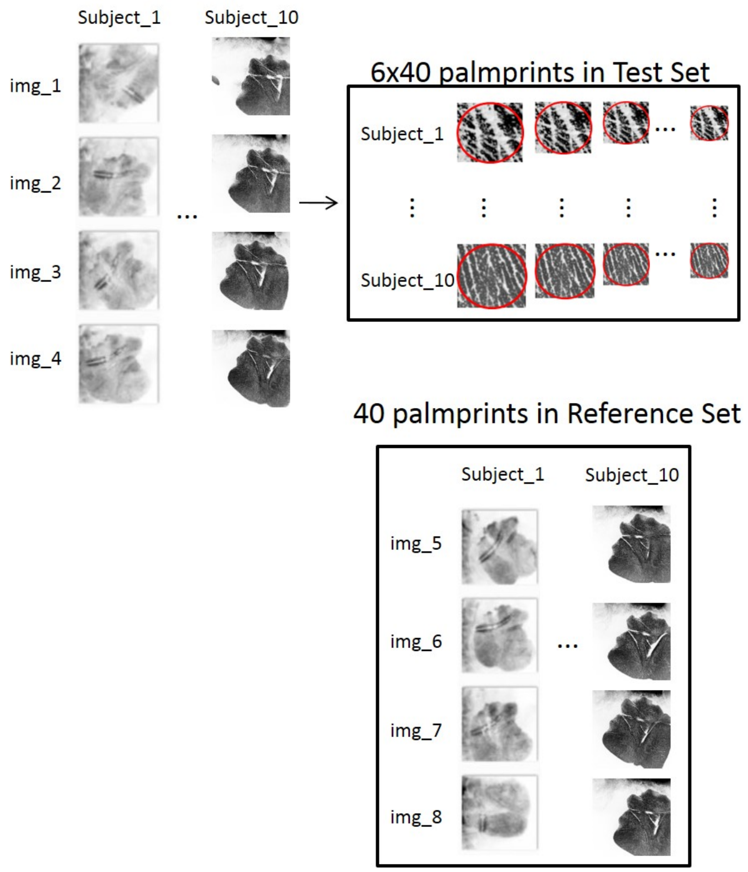


Figure 9. 4. Scheme of the organisation of "Tarragona Palmprint PF" database.

-*Tarragona Sagrada Familia PF* 2*D*: This database consists of 364 pictures obtained from [13] of 480 × 178 pixels taken from all around the Sagrada Familia cathedral in Barcelona, Spain. These images are stored in a sequential order, thus it is known that the first image was taken right next to the second image in terms of proximity, and so on. We have extracted the 800 most important features from the top half of each image using the SURF detector and extractor [3]. The reference set is composed of the even numbered images, while the test set is composed of patches extracted from the densest area (in terms of salient points) of the odd numbered images. More precisely, from each even image, we selected 9 circular patches with radius $r = 20,30, \ldots ,100$ pixels. The centres of the patches are the centres of masses of the salient points of the original images.

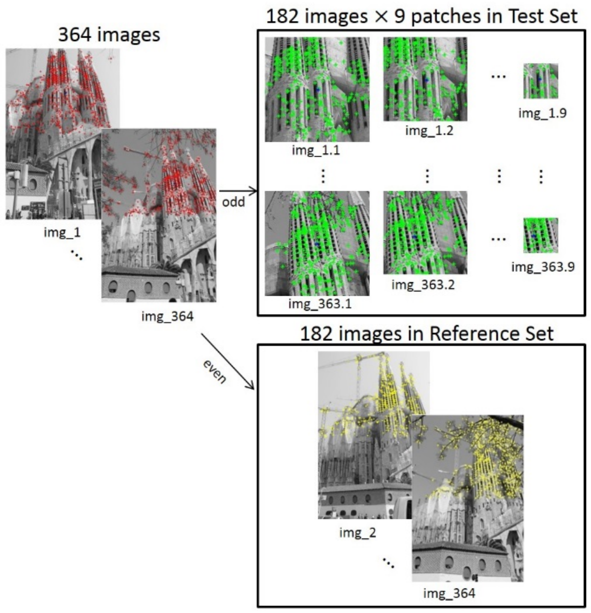


Figure 9. 5. Scheme of the organisation of "Tarragona Sagrada Familia PF 2D" database.

The code that generates these databases is available in [14].

9.5.2 Validation of the Matching module using a Palmprint Database

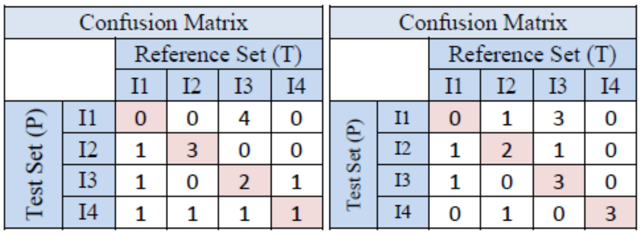
Since every module of the framework is independent of the application, the only modules that have to be validated are the point extractor and the matching algorithm. For the first one, the authors of the original database have already stated that salient points are extracted more optimally using the minutiae extractor that they presented in [4] and [5]. Therefore, we propose a comparison between two different matching algorithms in the *Matching* module: the FBP method [15] (given the efficiency that has been demonstrated in previous chapters) and the Hough method [16] which has been used for both biometric [1,4,5,15,17] and general image registration [6,18].

We compare every circular patch with every image on the reference set, thus a total of $40 \times 40 \times 7 = 11'200$ comparisons (1'600 per radius) are computed. This process is repeated for the two options of *Matching* module proposed. For this experimental validation, we employed the minutiae cost $c_{minutiae}$ definitions explained in chapter 2, this time using the partial palmprint P and the candidate partial palmprint F_a as the inputs and setting $\lambda_c = 0.85$ as the weighting parameter between the angular distance and the

Euclidean distance. Given the output partial palmprint P and the candidate partial palmprint F_a, the distance between them is defined as,

$$D(P, F_a) = \min_{\forall f_a} \frac{\sum_{m_i^P} c_{minutiae}\left(m_i^P, m_{fa(i)}^{F_a}\right)}{|P|} \tag{9.1}$$

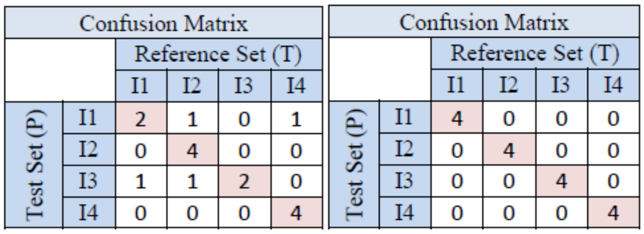
To show synthesised results, instead of presenting the whole table of comparisons between each circular patch and the whole reference set, we present the confusion matrices for the first four subjects of the database according to different radius sizes. Figures 9.6, 9.7, 9.8 and 9.9 show the confusion matrices for both frameworks, using $r = 100$, $r = 200$, $r = 400$ and $r = 600$. The classification is done between four different patches derived from different PI_i partial images with four elements contained on the reference set TI_i. With the previous knowledge that $PI_i = TI_i$, we then consider a successful identification when the elements on the diagonal possess the highest similarity s of the whole row, where $s = 1/d$. Notice that the functionality of the *PF-Registration* framework is enhanced when using the FBP method, which allows spurious mapping rejection.



Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	0	0	4	0
	I2	1	3	0	0
	I3	1	0	2	1
	I4	1	1	1	1

Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	0	1	3	0
	I2	1	2	1	0
	I3	1	0	3	0
	I4	0	1	0	3

Figure 9. 6. Confusion matrices for $r = 100$ pixels (left) and $r = 200$ pixels (right) using the FBP matching algorithm.



Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	2	1	0	1
	I2	0	4	0	0
	I3	1	1	2	0
	I4	0	0	0	4

Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	4	0	0	0
	I2	0	4	0	0
	I3	0	0	4	0
	I4	0	0	0	4

Figure 9. 7. Confusion matrices for $r = 400$ pixels (left) and $r = 600$ pixels (right) using the FBP matching algorithm.

Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	2	2	0	0
	I2	2	2	0	0
	I3	3	1	0	0
	I4	0	4	0	0

Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	1	3	0	0
	I2	1	3	0	0
	I3	1	3	0	0
	I4	0	3	0	1

Figure 9. 8. Confusion matrices for $r = 100$ pixels (left) and $r = 200$ pixels (right) using the Hough matching algorithm.

Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	1	3	0	0
	I2	0	4	0	0
	I3	0	2	2	0
	I4	0	2	0	2

Confusion Matrix					
		Reference Set (T)			
		I1	I2	I3	I4
Test Set (P)	I1	2	2	0	0
	I2	0	4	0	0
	I3	0	1	3	0
	I4	0	1	0	3

Figure 9. 9. Confusion matrices for $r = 400$ pixels (left) and $r = 600$ pixels (right) using the Hough matching algorithm.

Figure 9.10 shows the comparison of the recognition ratio obtained with the different options of the *Matching* module. This metric reflects the percentage of times a circular patch is correctly matched with one of the image of the same individual on the reference set. Notice the use of FBP derives in a higher recognition rate than when using the Hough matching algorithm for small images. Most notably, as the input image ratio is of about 100 pixels, the increase in recognition is around 21%. It must be pointed out that when $r > 700$ pixels, both methods tend to have the same performance, arriving to the 100% recognition rate.

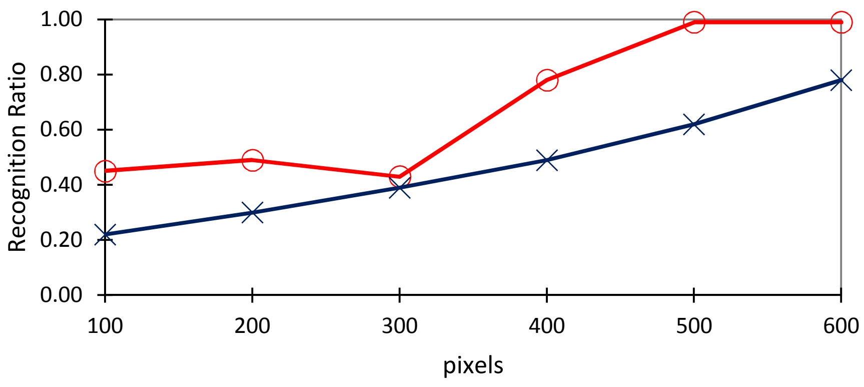
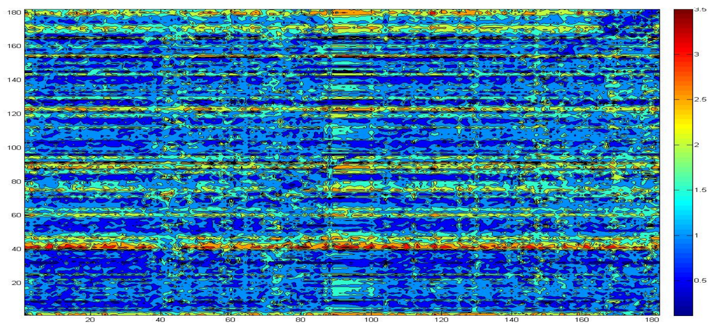


Figure 9. 10. Recognition ratio for both proposals as the radius of the circular patch increases; (O) *PF-Registration*, (×) Hough.

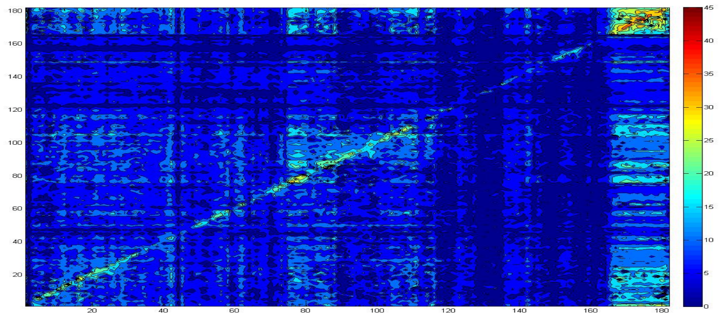
9.5.3 Neighbour Image Recognition using an Outdoors Scene Image Database

The aim of this experiment is to verify if the *PF-Registration* framework is able to deliver a D_a and a f_a that allow us to identify a patch n_{test} of an image n as a neighbour of images $n \pm 1$ (images $n_{reference}$ and $n_{reference} - 1$ in the reference set). To do so, we compared every circular patch in the test set with every image on the reference set, thus computing a total of $182 \times 182 \times 9 = 298'116$ correspondences in total (33'124 per radius). Afterwards, three matrices are created: a) a matrix with the distance values, b) a matrix containing the number of inlier mappings found per correspondence and c) a matrix calculated as the ratio of the previous two matrices. Since we know that image n_{test} from the test set is the neighbour of images $n_{reference}$ and $n_{reference} - 1$ from the reference set, it is desired to observe lower values in the diagonal of matrices a and c, as well as high values in the diagonal of matrix b. Based on the results obtained from the previous validation, the FBP algorithm [13] was used in the *Matching* module.

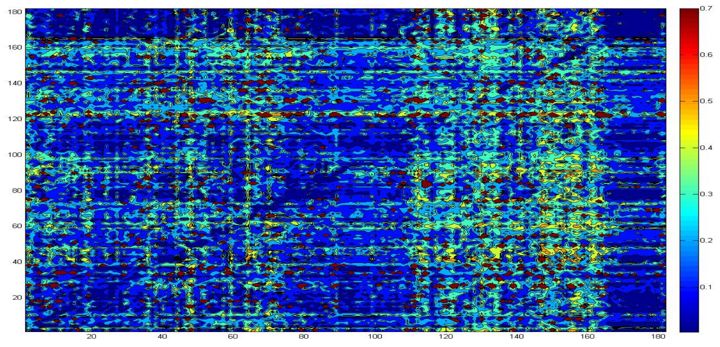
Figure 9.11 shows the obtained matrices for the patches of radius $r = 100$ pixels of the test set. Although figure 9.11.a. shows that the diagonal of matrix a has lower distance values, clearly the distance alone cannot be considered as a good metric to classify these patches. That is because there is a large variability on the number of salient points in the patches. In a similar way, figure 9.11.b shows the number of mappings between patches and images. Notice that the FBP algorithm has the ability to discard outliers, and thus, the inliers mappings result in a better form to identify a neighbour image. In this case, the diagonal seems to be more visible. Finally, figure 9.11.c shows the obtained distance (figure 9.11.a) normalised by the number of inliers (figure 9.11.b). That is, we want to know the quality of the obtained inlier mappings, and we do not want the number of outliers to influence on the metric value. In this final metric, the diagonal is also clearly visible and thus, this metric is useful as well.



a)



b)



c)

Figure 9. 11. Three different metrics used to evaluate the quality of the tiny to full matching algorithm. a) Distance, b) Number of inliers and c) Distance over number of inliers.

Once that the third metric has been selected as the most suitable to recognise neighbours, figure 9.12 shows the percentage of images for which our classification technique has selected at least one of the correct neighbour given all patch sizes and the full to full image comparison (expressed with the radius $r = 200$ pixels). Together with the original results, we have performed two additional tests in which only a portion of the salient points from the full image are used as input of the *PF-Registration* framework: 50

best salient points used and 100 best salient points used. Considering that the random chance of obtaining the correct result is $\frac{2}{182} \cong 0{,}011$, and that the sample sizes are tiny (e.g. a 20 pixel radius is approximately 5-10% of the full image), we believe that our reported accuracy is good given this high quality and real image database with no previously established ground truth. Moreover, the matching process is done between the patch of one image and a full image taken from another angle and position. Also, it is interesting to note that for $r \geq 60$, there is no need to use the whole set of 800 salient points from the full image to obtain a good accuracy.

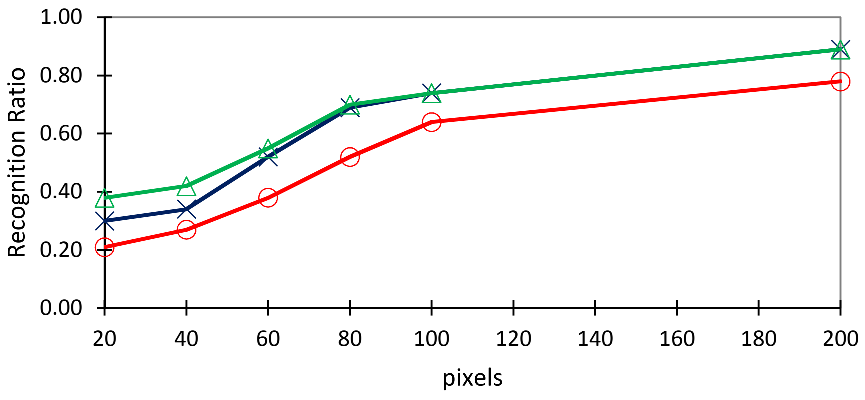


Figure 9. 12. Recognition ratio of the classification technique as the radius of the circular patch increases, given different number of salient points used from the full image; (O) 50 salient points, (×) 100 salient points, (Δ) 800 salient points.

Reducing the number of salient points used from the whole image has a direct impact on the runtime. In table 9.1, we show the time spent to compute the comparison of one partial image against the whole reference set measured in seconds. In terms of runtime, the difference between using 100 salient points of the full image compared to 800 salient points is considerable. For these tests, we used a PC with Intel 3.4 GHz CPU and Windows 7 operating system.

Table 9. 1. Runtime (in seconds) to compare a patch against the whole reference set.

Radius (pixels)	20	40	60	80	100	200
50 best salient points	853.41	876.85	902.24	904.24	910.34	910.39
100 best salient points	917.64	921.18	925.65	946.02	994.32	1065.70
800 best salient points	951.37	988.78	1076.91	1999.37	2702.01	15458.47

9.6 Discussion

Several methods have been presented to solve the image registration problem. Some of them are general methods applicable to a large spectrum of problems, but other ones as in the case of palmprint recognition, are application dependent. In some scenarios, image registration is based on finding a partial patch of the image inside a larger one. In these cases, most of the typical image registration methods (general methods or application dependent ones) that do not consider this specific feature are not able to obtain optimal results. In this chapter, we have presented a non-application dependent method that specifically considers the case that one of the images is a small part of the other one. It is based on two main steps that involve a matching and voting process, and generates a correspondence, which in connection with the work that has been presented previously, could be later used in frameworks that aim to find a representative prototype.

In the experimental section, we have shown the functionality of the method in two completely different applications: palmprint recognition and outdoor scenes detection. The first one provides a comparison with a state of the art method specifically designed for PF matching, and the second one validates the framework in a database with no requirement of a ground truth. For both scenarios, FBP has proven to be the matching algorithm that achieves a high accuracy in a reduced runtime. Also, it is shown that the framework is capable of considering only a portion of the salient points detected and extracted from the full image to further reduce the runtime while maintaining the accuracy rate.

Chapter 10
Conclusions and Future Work

10.1 Conclusions

In this work, we have presented a new approach for aiding the matching process commonly addressed in pattern recognition and computer vision tasks, based on the use and manipulation of correspondences, which are the one-to-one relations between the elements of an output object and an input object. The objects that can be matched through correspondences can be very diverse (i.e. images, body parts, social networks, characters, chemical structures, etc.) and can be represented through different data structures commonly used in these fields, such as sets of points, strings, graphs, trees and data clusters.

We have devoted the first part of this work to clearly identify and define what correspondences are, providing a sufficient amount of information regarding their generation, properties, and the forms to measure the dissimilarity of two or more correspondences through the proper use of a distance function between them.

As our main contribution, we have introduced the concept of learning the consensus for correspondences, which is a framework that is specifically designed to use these correspondences to learn a final consensus correspondence which has a better accuracy than the original ones, based on several constraints which have been studied and described throughout this work.

In addition, we have presented two advances that could potentially aid and diversify the current consensus framework. On the one hand, a new distance function and a weighted mean search strategy for a pair of correspondences have been introduced; these two concepts could be useful not only for the consensus framework, but also for other tasks that are widely demanded in pattern recognition, such as classification, ensemble clustering, machine learning, among others. On the other hand, a specific framework has been developed for image registration, where the output image is a tiny representation of the input image. We considered pertinent to include this latter development in this work due to its evident relation with correspondences in general.

10.2 Future Work

From the developments and results presented in this work, the following tasks are considered our future perspectives:

- As show in chapter 8, a new edit distance for correspondences and a search strategy for weighted means has been proposed in the present work. Nonetheless, it has not yet been explored how these two advances could enhance the consensus framework presented in this document.
- In chapter 9, a parallel contribution developed during this research time was presented, where we specifically address the issue of an output image being matched to a larger input one, referred as the *PF-Registration* framework. It is possible that better results can be achieved by combining this framework with other proposals through the application of a partial to full consensus learning framework.
- More methodologies that have been previously presented in literature could be explored for learning the consensus of a set of correspondences. For instance, the embedding algorithm has proven to be a reliable method in other domains such as strings, graphs and data clusters. Moreover, numerous voting schemes and constraints related to diverse data representations could be used as well to design new consensus models.
- By establishing correspondences as reliable data structures for data representation, it is possible to explore more uses for correspondences. For instance, machine learning, kernels, ensemble clustering and classification are just some of the many options that can be developed in the future.
- The datasets presented for the validation of this work will be collected in order to present a repository specifically designed for matching and classification based on correspondences.
- Finally, it is imperative that the frameworks presented in this work find their way in aiding more areas of science, beyond the ones explored so far.

LIST OF PUBLICATIONS

This dissertation has led to the following communications:

Journals

1. C. F. Moreno-García and F. Serratosa, "Fast and efficient palmprint identification of a small sample within a full image," Computación y Sistemas, vol. 18, no. 4, pp: 683-691, DOI 10.13053/CyS-18-4-2059, 2014.
2. C. F. Moreno-García and F. Serratosa, "Consensus of two sets of correspondences through optimisation functions," Pattern Analysis and Applications, pp:1-13, DOI 10.1007/s10044-015-0486-y, 2015.
3. C. F. Moreno-García and F. Serratosa, "Online learning the consensus of multiple correspondences between sets," Knowledge-Based Systems, vol. 90, no. December 2015, pp: 49-57, DOI 10.1016/j.knosys.2015.09.034, 2015.
4. C. F. Moreno-García and F. Serratosa, "Consensus of multiple correspondences between sets of elements," Computer Vision and Image Understanding, vol. 142, no. January 2016, pp: 50–64, DOI 10.1016/j.cviu.2015.08.008, 2016.
5. C. F. Moreno-García, M. Aceves-Martins, and F. Serratosa, "Unsupervised machine learning application to perform a systematic review and meta-analysis in medical research," Computación y Sistemas, vol. 20, no. 1, pp: 7-17, DOI 10.13053/CyS-20-1-2360, 2016.

Submitted

6. C. F. Moreno-García, F. Serratosa, and X. Jiang, "Simultaneously computing weighted means of a pair of correspondences using the correspondence edit distance," submitted to: Pattern Recognition.
7. C. F. Moreno-García and F. Serratosa, "Obtaining the consensus of multiple correspondences between graphs through online learning," submitted to: Pattern Recognition Letters (Special Issue).

Conference Contributions

1. C. F. Moreno-García and F. Serratosa, "Weighted mean assignment of a pair of correspondences using optimisation functions," Syntactic and Structural Pattern Recognition (S+SPR 2014), LNCS 8621, pp. 301-311, Joensuu, Finland.
2. C. F. Moreno-García, X. Cortés, and F. Serratosa, "Partial to full image registration based on candidate positions and multiple correspondences," The 20th Iberoamerican Congress on Pattern Recognition (CIARP 2014), LNCS 8827, pp. 745-753, Puerto Vallarta, Mexico.
3. C. F. Moreno-García, F. Serratosa, and X. Cortés, "Consensus of two graph correspondences through a generalization of the bipartite graph matching," Graph Based Representations and Pattern Recognition (GbR 2015), LNCS 9069, pp: 87-97, Beijing, China.
4. C. F. Moreno-García, F. Serratosa, and X. Cortés, "Iterative versus voting method to reach consensus given multiple correspondences of two sets," Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2015), LNCS 9117, pp: 530-540, Santiago de Compostela, Spain.

Submitted

5. C. F. Moreno-García, X. Cortés, and F. Serratosa, "Generalised median of a set of correspondences based on the hamming distance," submitted to: International Conference on Patter Recognition (ICPR 2016), Cancun, Mexico.
6. C. F. Moreno-García, X. Cortés, and F. Serratosa, "A graph repository for learning error-tolerant graph matching," submitted to: Syntactic and Structural Pattern Recognition (S+SPR 2016), Merida, Mexico.

Local Workshops

1. F. Serratosa, X. Cortés, and C. F. Moreno-García, "Aprenentatge de costos per la comparació de grafs," 2nd Workshop on Graph-Based Technologies and Applications (GraphTA 2014), Barcelona, Spain.
2. C. F. Moreno-García, X. Cortés, and F. Serratosa, "Consensus based methodologies for image labelling using optimization functions," The 1st Doctoral Workshop in Computer Science and Mathematics (DCSM 2014), Tarragona, Spain.
3. C. F. Moreno-García, "Consensus calculation of multiple input correspondences," The 2nd Doctoral Workshop in Computer Science and Mathematics (DCSM 2015), Tarragona, Spain.
4. C. F. Moreno-García, "Improving Image Recognition using Correspondences," Jornadas de Cooperación Conacyt-Cataluña 2016 (JCCC 2016), Barcelona Tech, Spain.

REFERENCES

Chapter 1

1. H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," Comput. Vis. Image Underst., vol. 89, no. 2–3, pp: 114–141, 2003.
2. M. Izadi and P. Saeedi, "Robust weighted graph transformation matching for rigid and nonrigid image registration," IEEE Trans. Image Process., vol. 21, no. 10, pp: 4369–4382, 2012.
3. G. Sanromá, R. Alquézar, and F. Serratosa, "A new graph matching method for point-set correspondence using the EM algorithm and Softassign," Comput. Vis. Image Underst., vol. 116, no. 2, pp: 292–304, 2012.
4. G. Sanroma, A. Penate-Sanchez, R. Alquézar, F. Serratosa, F. Moreno-Noguer, J. Andrade-Cetto, and M. Á. González Ballester, "MSClique: Multiple structure discovery through the maximum weighted clique problem," PLoS One, vol. 11, no. 1, p. e0145846, 2016.
5. J. Yan, M. Cho, and H. Zha, "Multi-graph matching via affinity optimization with graduated consistency regularization," IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, 2016.
6. A.K. Jain, P. Flynn, and A.A. Ross, "Handbook of biometrics," Springer, 2009.
7. M. Prätorius, A. Scherzinger, and K. Hinrichs, "SkInteract: An on-body interaction system based on skin-texture recognition," International Federation for Information Processing, 2015, vol. 9299, pp: 425–432.
8. D. Peralta, M. Galar, I. Triguero, D. Paternain, S. García, E. Barrenechea, J. M. Benítez, H. Bustince, and F. Herrera, "A survey on fingerprint minutiae-based local matching for verification and identification: Taxonomy and experimental evaluation," Inf. Sci. (NY)., vol. 315, pp: 67–87, 2015.
9. F. Yue, W.-M. Zuo, and D. P. Zhang, "Survey of palmprint recognition algorithms," Acta Autom. Sin., vol. 36, no. 3, pp: 353–365, 2010.
10. D. Zhang, W.-M. Zuo, and F. Yue, "A comparative study of palmprint recognition algorithms," ACM Comput. Surv., vol. 44, no. 1, pp: 1–37, 2012.
11. S. Agarwal and P. C. Gupta, "Identification of human through palmprint: A review," vol. 1, no. 10, pp: 19–22, 2012.
12. P. Somvanshi and M. Rane, "Survey of palmprint recognition," International Journal of Scientific & Engineering Research, vol. 3, no. 2, 2012.
13. W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," ACM Comput. Surv., vol. 35, no. 4, pp: 399–458, 2003.
14. M. Chli and A. Davison, "Active matching," Comput. Vision ECCV 2008, pp: 72–85, 2008.
15. F. Serratosa, R. Alquézar, and N. Amézquita, "A probabilistic integrated object recognition and tracking framework," Expert Syst. Appl., vol. 39, no. 8, pp: 7302–7318, 2012.
16. F. Serratosa, X. Cortés, and A. Solé, "Component retrieval based on a database of graphs for hand-written electronic-scheme digitalisation", Expert Systems with Applications, vol. 40, pp: 2493-2502, 2013.
17. K. Riesen and H. Bunke, "IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning," SSPR, pp: 287–297, 2008.
18. P. Willett, "Similarity-based virtual screening using 2D fingerprints.," Drug Discov. Today, vol. 11, no. 23–24, pp: 1046–53, 2006.

19. E. Duesbury, J. Holliday, and P. Willett, "Maximum common substructure-based data fusion in similarity searching," J. Chem. Inf. Model., vol. 55, no. 2, pp: 222–230, 2015.
20. A. Solé-Ribalta, C. Granell, S. Gómez, and A. Arenas, "Information transfer in community structured multiplex networks," Front. Phys., vol. 3, p. 13, 2015.
21. X. Wang, S. Liu, J. Liu, J. Chen, J. Zhu, and B. Guo, "A Full Picture of Relevant Topics," IEEE Trans. Vis. Comput. Graph., vol. 2626, no. c, pp: 1–1, 2016.
22. B. Zitová and J. Flusser, "Image registration methods: A survey," Image Vision Computing, vol. 21, no. 11, pp: 977-1000, 2003.
23. A.A. Goshtasby, "2-D and 3-D image registration for medical, remote sensing, and industrial applications," Wiley Press, 2005.
24. J. Salvi, C. Matabosch, D. Fofi, and J. Forest, "A review of recent range image registration methods with accuracy evaluation," Image Vision Comput. vol. 25, no. 5, pp: 578-596, 2007.
25. G. Navarro, "A guided tour to approximate string matching". ACM Computing Surveys, vol. 33, no. 1, pp: 31–88, 2001.
26. D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," International Journal of Pattern Recognition and Artificial Intelligence, vol. 18, no. 3, pp: 265-298, 2004.
27. P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last ten years". International Journal of Pattern Recognition and Artificial Intelligence, vol. 28, no. 1, 2015.
28. M. Vento, "A long trip in the charming world of graphs for pattern recognition," Pattern Recognit., vol. 48, no. 2, pp: 291–301, Feb. 2015.

Chapter 2

1. W.W. Cohen, P.D. Ravikumar, and S.E. Fienberg, "A comparison of string distance metrics for name-matching tasks," IIWeb, vol. 2003, pp: 73-78. 2003.
2. S. H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," International Journal of Mathematical Models and Methods in Applied Sciences, 2007.
3. G. Navarro, "A guided tour to approximate string matching". ACM Computing Surveys, vol. 33, no. 1, pp: 31–88, 2001.
4. V.I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," Soviet Physics Doklady, Cybernetics and Control Theory, vol. 10, pp: 707-710, 1966.
5. P. Bille, "A survey on tree edit distance and related problems," Theoretical Computer Science, vol. 337, no. 9, pp: 217-239, 2005.
6. A. Sanfeliu and K.S. Fu, "A distance measure between attributed relational graphs for pattern recognition," IEEE Transactions on Systems, Man, and Cybernetics, vol. 13, no. 3, pp: 353-362, 1983.
7. X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," Pattern Anal. Appl., vol 13, no. 1, pp: 113-129, 2010.
8. A. Solé-Ribalta, F. Serratosa, and A. Sanfeliu, "On the graph edit distance cost: Properties and applications," Int. J. Pattern Recognit. Artif. Intell., vol. 26, no. 05, 1260004, 2012.
9. F. Serratosa, "Computation of graph edit distance: Reasoning about optimality and speed-up," Image Vis. Comput., vol. 40, pp. 38–48, 2015.

10. F. Serratosa and X. Cortés, "Interactive graph-matching using active query strategies," Pattern Recognition, vol. 48, no. 4, pp: 1364-1373, 2015.
11. M. Ferrer, E. Valveny, and F. Serratosa, "Median graph computation by means of a genetic approach based on minimum common supergraph and maximum common subgraph," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5524, pp: 346–353, 2009.
12. A. Solé-Ribalta and F. Serratosa, "Graduated assignment algorithm for finding the common labelling of a set of graphs," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6218, pp: 180–190, 2010.
13. S. H. Amiri and M. Jamzad, "Efficient multi-modal fusion on supergraph for scalable image annotation," Pattern Recognit., vol. 48, no. 7, pp: 2241–2253, 2015.
14. E. Duesbury, J. Holliday, and P. Willett, "Maximum common substructure-based data fusion in similarity searching," J. Chem. Inf. Model., vol. 55, no. 2, pp: 222–230, 2015.
15. H. Bunke, X. Jiang, and A. Kandel, "On the minimum common supergraph of two graphs," Computing, vol. 25, pp: 13–25, 2000.
16. B. Rao and A. Mitra, "An approach to merging of two community subgraphs to form a community graph using graph mining techniques," 2014 IEEE Int. Conf. Comput. Intell. Comput. Res. IEEE ICCIC 2014, pp: 13–23, 2015.
17. R. Burkard, M. Dell'Amico, and S. Martello, "Assignment Problems," Society for Industrial and Applied Mathematics, Philadelphia, 2009.
18. H.W. Kuhn, "The hungarian method for the assignment problem export," Naval Research Logistics Quarterly, vol. 2, no. 1-2, pp: 83–97, 1955.
19. J. Munkres, "Algorithms for the assignment and transportation problems," Journal of the Society of Industrial and Applied Mathematics, vol. 5, no. 1, pp: 32–38, 1957.
20. R. Jonker, and T. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," Computing, vol. 38, no. 4, pp: 325–340, 1987.
21. C. Tomasi, and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMUCS-91-132, Carnegie Mellon University, 1991.
22. W. Han, and M. Brady, "Real-time corner detection algorithm for motion estimation," Image and Vision Computing, pp: 695–703, 2005.
23. E. Rosten, and T. Drummond, "Machine learning for high-speed corner detection," European Conference on Computer Vision, vol. 1, pp: 430–443, 2006.
24. K. Mikolajczyk, and C. Schmid, "A performance evaluation of local descriptors," IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 10, pp: 1615–1630, 2005.
25. B. Zitová and J. Flusser, "Image registration methods: A survey," Image Vision Computing, vol. 21, no. 11, pp: 977-1000, 2003.
26. A.A. Goshtasby, "2-D and 3-D image registration for medical, remote sensing, and industrial applications," Wiley Press, 2005.
27. J. Salvi, C. Matabosch, D. Fofi, and J. Forest, "A review of recent range image registration methods with accuracy evaluation," Image Vision Comput. vol. 25, no. 5, pp: 578-596, 2007.
28. M. Kashif, T. M. Deserno, D. Haak, and S. Jonas, "Feature description with SIFT, SURF, BRIEF, BRISK, or FREAK? A general question answered for bone age assessment," Comput. Biol. Med., vol. 68, pp. 67–75, 2016.

29. E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, pp: 105–119, 2010.
30. C. Harris and M. Stephens, "A combined corner and edge detector," Proceedings of the 4th Alvey Vision Conference, pp: 147–151, 1988.
31. J. Shi and C. Tomasi, "Good features to track," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp: 593–600. 1994.
32. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," Computer Vision and Image Understanding (CVIU), vol. 110, no. 3, pp: 346-359, 2008.
33. D.G. Lowe, "Distinctive image features from scale-invariant keypoints," IJCV vol. 60, no. 2, pp: 91–110. 2004.
34. Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," Int. J. Comput. Vision, vol. 13, no.2, pp: 119–152, 1994.
35. M.A. Fischler "Random Sample Consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Commun. ACM, vol. 24, no. 6, pp: 381–395, 1981.
36. D.H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," IEEE Trans. on Pattern Analysis and Matching Intelligence, 1980.
37. A.A. Kassim, T. Tan, and K.H. Tan, "A comparative study of efficient generalised Hough transform techniques," Image and Vision Computing vol. 17, pp: 737–748, 1999.
38. A.K. Jain, P. Flynn, and A.A. Ross, "Handbook of biometrics," Springer, 2009.
39. G. Sanromà, R. Alquézar, F. Serratosa, and B. Herrera, "Smooth point-set registration using neighbouring constraints," Pattern Recognition Letters, vol. 33, pp: 2029-2037, 2012.
40. E. Kokiopoulou, and P. Frossard, "Graph-based classification of multiple observation sets," Pattern Recognition, vol. 43, pp: 3988–3997, 2010.
41. L. Chang, M. Arias-Estrada, J. Hernández-Palancar, and L. Enrique-Sucar, "Partial shape matching and retrieval under occlusion and noise," CIARP 2014, LNCS 8827 Springer, pp: 151-158, 2014.
42. C. Wachinger and N. Navab, "Simultaneous registration of multiple images: Similarity metrics and efficient optimization," IEEE Trans. on Pattern Analysis and Matching Intelligence, vol. 35, no. 5, pp: 1221-1233, 2013.
43. J. Zhao, J. Ma, J. Tian, J. Ma, and D. Zhang, "A robust method for vector field learning with application to mismatch removing," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., no. Lic, pp: 2977–2984, 2011.
44. S.K. Dewan, "Elementary, Watson - scan a palm, find a clue," The New York Times, November 2003. Available in http://www.nytimes.com/2003/11/21/nyregion/elementary-watson-scan-a-palm-find-a-clue.html?pagewanted=all
45. F. Yue, W.M. Zuo, and D. P. Zhang, "Survey of palmprint recognition algorithms," Acta Autom. Sin., vol. 36, no. 3, pp: 353–365, 2010.
46. D. Zhang, W. M. Zuo, and F. Yue, "A comparative study of palmprint recognition algorithms," ACM Comput. Surv., vol. 44, no. 1, pp: 1–37, 2012.
47. S. Agarwal and P. C. Gupta, "Identification of human through palmprint: A review," International Journal of Advanced Research in Computer Engineering & Technology, vol. 1, no. 10, pp: 19-22, 2012.

48. P. Somvanshi and M. Rane, "Survey of palmprint recognition," International Journal of Scientific & Engineering Research, vol. 3, no. 2, 2012.
49. W.K. Kong and D. Zhang, "Using low-resolution palmprint Images and Texture Analysis for Personal Identification," ICPR, 2002.
50. J. Funada, N. Ohta, M. Mizoguchi, T. Temma, K. Nakanishi, A. Murai, T. Sugiuchi, T. Wakabayashi, and Y. Yamada, "Feature Extraction Method for Palmprint Considering Elimination of Creases," Proc. 14th Int. Conf. Pattern Recognition, pp: 1849-1854, 1998.
51. A.K. Jain and M. Demirkus, "On latent palmprint matching," MSU Technical Report, 2008.
52. A.K. Jain and J. Feng, "Latent palmprint matching," IEEE Trans. on PAMI, 2009.
53. J. Dai and J. Zhou, "Multifeature-based high-resolution palmprint recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 33, no. 5, pp: 945-957, 2011.
54. J. Dai, J. Feng, and J. Zhou, "Robust and efficient ridge based palmprint matching," IEEE Transactions on Pattern Analysis and Matching Intelligence, vol. 34, no. 8, 2012.
55. X. Wang, J. Liang, and M. Wang, "On-line fast palmprint identification based on adaptive lifting wavelet scheme," Knowledge-Based Systems, vol. 42, pp: 68-73, 2013.
56. O. Nibouche, J. Jiang, and P. Trundle, "Analysis of performance of palmprint matching with enforced sparsity," Digital Signal Processing, vol. 22, no. 2, pp: 348-355, 2012.
57. I.T. Jolliffe, "Principal component analysis," Second Edition, Springer, 2002.
58. G.S. Badrinath and P. Gupta, "Palmprint based recognition system using phase-difference information," Future Generation Computer Systems, vol. 28, no. 1, pp: 287-305, 2012.
59. X. Wang, L. Lei, and M. Wang, "Palmprint verification based on 2D-gabor wavelet and pulse-coupled neural network" Knowledge-Based Systems, vol. 27, pp: 451-455, 2012.
60. S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," IEEE Trans. Pattern Anal. Mach. Intell., vol. 18, no. 4, pp. 377–388, 1996.
61. L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "Subgraph Transformations for the Inexact Matching of Attributed Relational Graphs," in Graph Based Representations in Pattern Recognition, 1998, pp. 43–52.
62. D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," International Journal of Pattern Recognition and Artificial Intelligence, vol. 18, no. 3, pp: 265-298, 2004.
63. P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last ten years". International Journal of Pattern Recognition and Artificial Intelligence, vol. 28, no. 1, 2015.
64. M. Vento, "A long trip in the charming world of graphs for Pattern Recognition," Pattern Recognit., vol. 48, no. 2, pp: 291–301, Feb. 2015.
65. H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," Pattern Recognit. Lett., vol. 1, no. 4, pp: 245–253, 1983.
66. H. Bunke, "On a relation between graph edit distance and maximum common subgraph," Pattern Recognit. Lett., vol. 18, no. 8, pp: 689–694, 1997.

67. K. Riesen and H., Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," Image Vision Comput, vol. 27, no. 7, pp: 950-959, 2009.
68. F. Serratosa, "Fast computation of bipartite graph matching," Pattern Recognition Letters, vol. 45, pp: 244-250, 2014.
69. F. Serratosa, "Speeding up fast bipartite graph matching through a new cost matrix," International Journal of Pattern Recognition and Artificial Intelligence, vol. 29, no. 2, 2015.
70. K. Riesen, H. Bunke, and A. Fischer, "Improving graph edit distance approximation by centrality measures," Proc. - Int. Conf. Pattern Recognit., pp. 3910–3914, 2014.
71. F. Serratosa and X. Cortés, "Graph edit distance: Moving from global to local structure to solve the graph-matching problem," Pattern Recognit. Lett., vol. 65, pp. 204–210, 2015.
72. H. Bunke, S. Günter, and X. Jiang, "Towards bridging the gap between statistical and structural pattern recognition: Two new concepts in graph matching," ICAPR, pp: 1–11, 2001.
73. H. Bunke, X. Jiang, K. Abegglen, and A. Kandel, "On the weighted mean of a pair of strings," Pattern Analysis and Applications, vol. 5, pp: 23-30, 2002.
74. H. Bunke, and S. Gunter, "Weighted mean of a pair of graphs", Computing, vol. 67, pp: 209-224, 2001.
75. L. Franek, X. Jiang, and C. He, "Weighted mean of a pair of clusterings," Pattern Analysis and Applications, vol. 17, pp: 153-166, 2014.
76. X. Jiang, A. Munger, and H. Bunke, "On median graphs: properties, algorithms, and applications," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 10, pp: 1144-1151, 2001.
77. T. Kohonen, "Median Strings," Pattern Recognition Letters, vol. 3, pp: 309-3013. 1985.
78. X. Jiang and H. Bunke, "Learning by generalized median concept," Pattern Recognition and Machine Vision, P. Wang (Ed.), River Publishers, Aalborg, pp 1–16, 2010.
79. R. Raveaux, S. Adam, P. Héroux, and É. Trupin, "Learning graph prototypes for shape recognition," Comput. Vis. Image Underst., vol. 115, no. 7, pp: 905–918, 2011.
80. M. Ferrer, E. Valveny, F. Serratosa, and H. Bunke, "Graph-based k -means clustering: A comparison of the set median versus the generalized median graph," CAIP, pp: 342–350, 2009.
81. S. Vega-Pons and J. Ruiz-Shulcloper, "A survey of clustering ensemble algorithms," IJPRAI, vol. 25, no. 3, pp: 337-372, 2011.
82. R. Ghaemi, N. Sulaiman, H. Ibrahim, and N. Mustapha, "A survey: Clustering ensembles techniques," Eng. Technol., vol. 38, no. February, pp: 636–645, 2009.
83. J. Abreu and J. R. Rico-Juan, "An improved fast edit approach for two-string approximated mean computation applied to OCR," Pattern Recognit. Lett., vol. 34, no. 5, pp. 496–504, 2013.
84. J. Oh, N. Kwak, M. Lee, and C.H. Choi, "Generalized mean for feature extraction in one-class classification problems," Pattern Recognit., vol. 46, no. 12, pp: 3328–3340, 2013.
85. X. Jiang and H. Bunke, "Optimal lower bound for generalized median problems in metric space," Struct. Syntactic, Stat. Pattern Recognit., pp: 143–151, 2002.

86. J. Abreu and J. R. Rico-Juan, “A new iterative algorithm for computing a quality approximate median of strings based on edit operations,” Pattern Recognit. Lett., vol. 36, no. 1, pp. 74–80, 2014.
87. X. Jiang, J. Wentker, and Ferrer, “Generalized median string computation by means of string embedding in vector spaces,” Pattern Recognition Letters, vol. 33, pp: 842-852, 2012.
88. M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, and H. Bunke, “Generalized median graph computation by means of graph embedding in vector spaces,” Patter Recognition, vol. 43, pp: 1642-1655, 2010.
89. M. Ferrer, D. Karatzas, E. Valveny, I. Bardaji, and H. Bunke, “A generic framework for median graph computation based on a recursive embedding approach,” Comput. Vis. Image Underst., vol. 115, no. 7, pp: 919–928, 2011.
90. L. Franek and X. Jiang, “Ensemble clustering by means of clustering embedding in vector spaces,” Pattern Recognition, vol. 47, pp: 833-842, 2014.
91. L. Franek and X. Jiang, “Evolutionary weighted mean based framework for generalized median computation with application to strings,” G. Gimel’ et al. (Eds.), SSPR & SPR 2012, LNCS 7626, pp: 70-78, 2012.
92. G. Saygili, L. Van Der Maaten, and E. A. Hendriks, “Adaptive stereo similarity fusion using confidence measures,” Comput. Vis. Image Underst., vol. 135, no. November, pp: 95–108, 2015.
93. A. Ross, A.K. Jain, and J. Reisman, “A hybrid fingerprint matcher,” Pattern Recognit., vol. 36, no. 7, pp: 1661–1673, 2003.
94. J. Gu, J. Zhou, and C. Yang, “Fingerprint recognition by combining global structure and local cues,” IEEE Trans. Image Process., vol. 15, no. 7, pp: 1952–1964, 2006.
95. W. Sheng, G. Howells, M. C. Fairhurst, F. Deravi, and K. Harmer, “Consensus fingerprint matching with genetically optimised approach,” Pattern Recognit., vol. 42, no. 7, pp: 1399–1407, 2009.
96. L. Liu, T. Jiang, J. Yang, and C. Zhu, “Fingerprint registration by maximization of mutual information,” IEEE Trans. Image Process., vol. 15, no. 5, pp: 1100–1110, 2006.
97. A.K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, “Filterbank-based fingerprint matching,” IEEE Trans. Image Process., vol. 9, no. 5, pp: 846–859, 2000.
98. C.H. Park, J.J. Lee, M.J.T. Smith, S. Il Park, and K. H. Park, “Directional filter bank-based fingerprint feature extraction and matching,” IEEE Trans. Circuits Syst. Video Technol., vol. 14, no. 1, pp: 74–85, 2004.
99. J. L. García-Lapresta, “Voting rules: From the ordinal setting to the linguistic approach,” 2012.
100. J. R. Rico-Juan and J. M. Iñesta, “Normalisation of confidence voting methods applied to a fast handwritten OCR classification,” Adv. Soft Comput., vol. 45, pp: 405–412, 2007.
101. S. Saha and A. Ekbal, “Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition,” Data & Knowledge Engineering, vol. 85, pp: 15-39, 2013.
102. F. Meng, X. Li, and J. Pei, “A feature point matching based on spatial order constraints bilateral-neighbour vote.,” IEEE Trans. Image Process., vol. 24, no. 11, pp: 4160–4171, 2015.

103. J. R. Rico-Juan and J. M. Iñesta, "Normalisation of confidence voting methods applied to a fast handwritten OCR classification," Comput. Recognit. Syst. 2, vol. 45, no. Advances in Sof Computing, pp. 405–412, 2007.
104. J. R. Rico-Juan and J. M. Iñesta, "Confidence voting method ensemble applied to off-line signature verification," Pattern Anal. Appl., vol. 15, no. 2, pp. 113–120, 2012.
105. H.G. Ayad and M.S. Kamel, "On voting-based consensus of cluster ensembles," Pattern Recognit., vol. 43, no. 5, pp: 1943–1953, 2010.
106. A. Iqbal, A. Moh'd, and Z. Khan, "Semi-supervised clustering ensemble by voting," arXiv Prepr. arXiv1208.4138, pp: 1–5, 2012.
107. R. Timofte and L. Van Gool, "Adaptive and weighted collaborative representations for image classification," Pattern Recognit. Lett., vol. 43, no. 1, pp: 127–135, 2014.
108. L. Zhang, M. Yang, and X. Feng, "Sparse representation or collaborative representation: Which helps face recognition?" Proc. IEEE Int. Conf. Comput. Vis., pp: 471–478, 2011.
109. T. Hofmann and J. Basilico, "Collaborative machine learning," From Integr. Publ. Inf. Syst. to Inf. Knowl. Environ., pp: 173–182, 2005.
110. X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Adv. Artif. Intell., vol. 3, no. Section 3, pp: 1–19, 2009.
111. T. Bogers and A. Van Den Bosch, "Collaborative and content-based filtering for item recommendation on social bookmarking websites," CEUR Workshop Proc., vol. 532, pp: 9–16, 2009.
112. F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in Recommender Systems Handbook, 2011, pp: 1–35.
113. Z. Ghahramani, "Unsupervised learning. Advanced lectures on machine learning," Springer-Verlag, pp: 72-112, 2004.
114. S. Monti, P. Tamayo, J.P. Mesirov, and T.R. Golub, "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," Machine Learning, vol. 52, no. 1-2, pp: 91-118, 2003.
115. S. Mimaroglu and E. Erdil, "Combining multiple clusterings using similarity graphs," Pattern Recognition, vol. 44, pp: 694-70, 2010.
116. L. Murino, C. Angelini, I. De Feis, G. Raiconi, and R. Tagliaferri, "Beyond classical consensus clustering: The least squares approach to multiple solutions," Pattern Recognition Letters, vol. 32, pp: 1604-1612, 2011.
117. Y. Dong and H. Zhang, "Multiperson decision making with different preference representation structures: A direct consensus framework and its properties," Knowledge-Based Systems, vol. 58, pp: 45-57, 2014.
118. J. Wu and F. Chiclana, "A social network analysis trust-consensus based approach to group decision-making problems with interval-valued fuzzy reciprocal preference relations," Knowledge-Based Systems, vol. 59, no. March, pp: 97-107, 2014.
119. Q. Zhang and S. Sun, "Multiple-view multiple-learner active learning," Pattern Recognit., vol. 43, no. 9, pp: 3113–3119, 2010.
120. L. Zhang, T. Li, and X. Xu, "Consensus model for multiple criteria group decision making under intuitionistic fuzzy environment," Knowledge-Based Systems, vol. 57, no. February, pp: 127-135, 2014.

121. Z. Gong, X. Xu, L. Li, and C. Xu, "Consensus modeling with nonlinear utility and cost constraints: A case study". Knowledge-Based Systems, vol. 88, no. November, pp: 210-222, 2015.
122. X. Xu, X. Zhong, X. Chen, and Y. Zhou, "A dynamical consensus method based on exit-delegation mechanism for large group emergency decision making". Knowledge-Based Systems, vol. 86, no. September, pp: 237-249, 2015.

Chapter 3

1. L. Franek and X. Jiang, "Evolutionary weighted mean based framework for generalized median computation with application to strings," G. Gimel' et al. (Eds.), SSPR & SPR 2012, LNCS 7626, pp: 70-78, 2012.
2. S. Saha and A. Ekbal, "Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition," Data & Knowledge Engineering, vol. 85, pp: 15-39, 2013.
3. R. Burkard, M. Dell'Amico, and S. Martello, "Assignment Problems," Society for Industrial and Applied Mathematics, Philadelphia, 2009.
4. H.W. Kuhn, "The hungarian method for the assignment problem export," Naval Research Logistics Quarterly, vol. 2, no. 1-2, pp: 83–97, 1955.
5. J. Munkres, "Algorithms for the assignment and transportation problems," Journal of the Society of Industrial and Applied Mathematics, vol. 5, no. 1, pp: 32–38, 1957.
6. R. Jonker, and T. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," Computing, vol. 38, no. 4, pp: 325–340, 1987.
7. X. Jiang, A. Munger, and H. Bunke, "On median graphs: properties, algorithms, and applications," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 10, pp: 1144-1151, 2001.

Chapter 4

1. H. Bunke, X. Jiang, K. Abegglen, and A. Kandel, "On the weighted mean of a pair of strings," Pattern Analysis and Applications, vol. 5, pp: 23-30, 2002.
2. H. Bunke, and S. Gunter, "Weighted mean of a pair of graphs", Computing, vol. 67, pp: 209-224, 2001.
3. L. Franek, X. Jiang, and C. He, "Weighted mean of a pair of clusterings," Pattern Analysis and Applications, vol. 17, pp: 153-166, 2014.
4. R. Burkard, M. Dell'Amico, and S. Martello, "Assignment Problems," Society for Industrial and Applied Mathematics, Philadelphia, 2009.
5. C. Papadimitriou and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity". Dover Publications. 1998.
6. S. Saha and A. Ekbal, "Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition," Data & Knowledge Engineering, vol. 85, pp: 15-39, 2013.
7. H.W. Kuhn, "The hungarian method for the assignment problem export," Naval Research Logistics Quarterly, vol. 2, no. 1-2, pp: 83–97, 1955.
8. J. Munkres, "Algorithms for the assignment and transportation problems," Journal of the Society of Industrial and Applied Mathematics, vol. 5, no. 1, pp: 32–38, 1957.

9. R. Jonker, and T. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," Computing, vol. 38, no. 4, pp: 325–340, 1987.
10. J. Dai and J. Zhou, "Multifeature-based high-resolution palmprint recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 33, no. 5, pp: 945-957, 2011.
11. J. Dai, J. Feng, and J. Zhou, "Robust and efficient ridge based palmprint matching," IEEE Transactions on Pattern Analysis and Matching Intelligence, vol. 34, no. 8, 2012.
12. N.K. Ratha, K. Karu, S. Chen, and A.K. Jain, "A real-time matching system for large fingerprint databases," IEEE Trans. Pattern Anal. Mach. Intell., vol. 18, no. 8, pp: 799–813, 1996.
13. http://www.featurespace.org
14. E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, pp: 105–119, 2010.
15. C. Harris and M. Stephens, "A combined corner and edge detector," Proceedings of the 4th Alvey Vision Conference, pp: 147–151, 1988.
16. J. Shi and C. Tomasi, "Good features to track," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp: 593–600. 1994.
17. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," Computer Vision and Image Understanding (CVIU), vol. 110, no. 3, pp: 346-359, 2008.
18. D.G. Lowe, "Distinctive image features from scale-invariant keypoints," IJCV vol. 60, no. 2, pp: 91–110. 2004.
19. http://es.mathworks.com/help/vision/ref/matchfeatures.html
20. F. Serratosa, "Fast computation of bipartite graph matching," Pattern Recognition Letters, vol. 45, pp: 244-250, 2014.
21. http://deim.urv.cat/~francesc.serratosa/databases/
22. K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," Image Vision Comput, vol. 27, no. 7, pp: 950-959, 2009.

Chapter 5

1. K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," Image Vision Comput, vol. 27, no. 7, pp: 950-959, 2009.
2. H.W. Kuhn, "The hungarian method for the assignment problem export," Naval Research Logistics Quarterly, vol. 2, no. 1-2, pp: 83–97, 1955.
3. J. Munkres, "Algorithms for the assignment and transportation problems," Journal of the Society of Industrial and Applied Mathematics, vol. 5, no. 1, pp: 32–38, 1957.
4. R. Jonker, and T. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," Computing, vol. 38, no. 4, pp: 325–340, 1987.
5. http://www.featurespace.org

6. E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, pp: 105–119, 2010.
7. C. Harris and M. Stephens, “A combined corner and edge detector,” Proceedings of the 4th Alvey Vision Conference, pp: 147–151, 1988.
8. J. Shi and C. Tomasi, “Good features to track,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp: 593–600. 1994.
9. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” Computer Vision and Image Understanding (CVIU), vol. 110, no. 3, pp: 346-359, 2008.
10. D.G. Lowe, “Distinctive image features from scale-invariant keypoints,” IJCV vol. 60, no. 2, pp: 91–110. 2004.
11. http://es.mathworks.com/help/vision/ref/matchfeatures.html
12. F. Serratosa, “Fast computation of bipartite graph matching,” Pattern Recognition Letters, vol. 45, pp: 244-250, 2014.
13. http://deim.urv.cat/~francesc.serratosa/databases/

Chapter 6

1. K. Riesen and H. Bunke, “Approximate graph edit distance computation by means of bipartite graph matching,” Image Vision Comput, vol. 27, no. 7, pp: 950-959, 2009.
2. http://www.featurespace.org
3. http://www.robots.ox.ac.uk/~vgg/research/affine/
4. E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, pp: 105–119, 2010.
5. C. Harris and M. Stephens, “A combined corner and edge detector,” Proceedings of the 4th Alvey Vision Conference, pp: 147–151, 1988.
6. J. Shi and C. Tomasi, “Good features to track,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp: 593–600. 1994.
7. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” Computer Vision and Image Understanding (CVIU), vol. 110, no. 3, pp: 346-359, 2008.
8. D.G. Lowe, “Distinctive image features from scale-invariant keypoints,” IJCV vol. 60, no. 2, pp: 91–110. 2004.
9. http://es.mathworks.com/help/vision/ref/matchfeatures.html
10. F. Serratosa, “Fast computation of bipartite graph matching,” Pattern Recognition Letters, vol. 45, pp: 244-250, 2014.
11. http://deim.urv.cat/~francesc.serratosa/databases/

Chapter 7

1. K. Riesen and H. Bunke, “Approximate graph edit distance computation by means of bipartite graph matching,” Image Vision Comput, vol. 27, no. 7, pp: 950-959, 2009.
2. F. Serratosa and A. Sanfeliu, “Signatures versus histograms: Definitions, distances and algorithms,” Pattern Recognition, vol. 39, no. 5, pp: 921–934. 2006.

3. H.W. Kuhn, "The hungarian method for the assignment problem export," Naval Research Logistics Quarterly, vol. 2, no. 1-2, pp: 83–97, 1955.
4. J. Munkres, "Algorithms for the assignment and transportation problems," Journal of the Society of Industrial and Applied Mathematics, vol. 5, no. 1, pp: 32–38, 1957.
5. R. Jonker, and T. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," Computing, vol. 38, no. 4, pp: 325–340, 1987.
6. http://www.featurespace.org
7. http://www.robots.ox.ac.uk/~vgg/research/affine/
8. E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, pp: 105–119, 2010.
9. C. Harris and M. Stephens, "A combined corner and edge detector," Proceedings of the 4th Alvey Vision Conference, pp: 147–151, 1988.
10. J. Shi and C. Tomasi, "Good features to track," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp: 593–600. 1994.
11. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," Computer Vision and Image Understanding (CVIU), vol. 110, no. 3, pp: 346-359, 2008.
12. D.G. Lowe, "Distinctive image features from scale-invariant keypoints," IJCV vol. 60, no. 2, pp: 91–110. 2004.
13. http://es.mathworks.com/help/vision/ref/matchfeatures.html
14. F. Serratosa, "Fast computation of bipartite graph matching," Pattern Recognition Letters, vol. 45, pp: 244-250, 2014.
15. http://deim.urv.cat/~francesc.serratosa/databases/

Chapter 8

1. D. Huang, J. Lai, and C. D. Wang, "Ensemble clustering using factor graph," Pattern Recognit., vol. 50, pp: 131–142, 2016.
2. H. Bunke, X. Jiang, K. Abegglen, and A. Kandel, "On the weighted mean of a pair of strings," Pattern Analysis and Applications, vol. 5, pp: 23-30, 2002.
3. H. Bunke, and S. Gunter, "Weighted mean of a pair of graphs", Computing, vol. 67, pp: 209-224, 2001.
4. L. Franek, X. Jiang, and C. He, "Weighted mean of a pair of clusterings," Pattern Analysis and Applications, vol. 17, pp: 153-166, 2014.
5. K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," Image Vision Comput, vol. 27, no. 7, pp: 950-959, 2009.

Chapter 9

1. A.K. Jain and M. Demirkus, "On latent palmprint matching," MSU Technical Report, 2008.
2. D.G. Lowe, "Distinctive image features from scale-invariant keypoints," IJCV vol. 60, no. 2, pp: 91–110. 2004.
3. A.K. Jain and J. Feng, "Latent palmprint matching," IEEE Trans. on PAMI, 2009.

4. J. Dai and J. Zhou, "Multifeature-based high-resolution palmprint recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 33, no. 5, pp: 945-957, 2011.
5. J. Dai, J. Feng, and J. Zhou, "Robust and efficient ridge based palmprint matching," IEEE Transactions on Pattern Analysis and Matching Intelligence, vol. 34, no. 8, 2012.
6. D.H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," IEEE Trans. on Pattern Analysis and Matching Intelligence, 1980.
7. A.A. Kassim, T. Tan, and K.H. Tan, "A comparative study of efficient generalised Hough transform techniques," Image and Vision Computing vol. 17, pp: 737–748, 1999.
8. A.A. Goshtasby, "2-D and 3-D image registration for medical, remote sensing, and industrial applications," Wiley Press, 2005.
9. B. Zitová and J. Flusser, "Image registration methods: A survey," Image Vision Computing, vol. 21, no. 11, pp: 977-1000, 2003.
10. Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," Int. J. Comput. Vision, vol. 13, no.2, pp: 119–152, 1994.
11. M.A. Fischler "Random Sample Consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Commun. ACM, vol. 24, no. 6, pp: 381–395, 1981.
12. H.W. Kuhn, "The hungarian method for the assignment problem export," Naval Research Logistics Quarterly, vol. 2, no. 1-2, pp: 83–97, 1955.
13. A. Penate-Sanchez, F. Moreno-Nogue, J. Andrade-Cetto, and F. Fleuret, "LETHA: Learning from high quality inputs for 3D pose estimation in low quality images," Proceedings of the International Conference on 3D Vision, 2014.
14. http://deim.urv.cat/~francesc.serratosa/sw/
15. F. Serratosa, "Fast computation of bipartite graph matching," Pattern Recognition Letters, vol. 45, pp: 244-250, 2014.
16. N.K. Ratha, K. Karu, S. Chen, and A.K. Jain, "A real-time matching system for large fingerprint databases," IEEE Trans. Pattern Anal. Mach. Intell., vol. 18, no. 8, pp: 799–813, 1996.
17. A.K. Jain, P. Flynn, and A.A. Ross, "Handbook of biometrics," Springer, 2009.
18. A. Zamberletti, I. Gallo, S. Albertini, and L. Noce, "Neural 1D Barcode Detection Using the Hough Transform," IPSJ Trans. Comput. Vis. Appl., vol. 7, pp: 1–9, 2014.