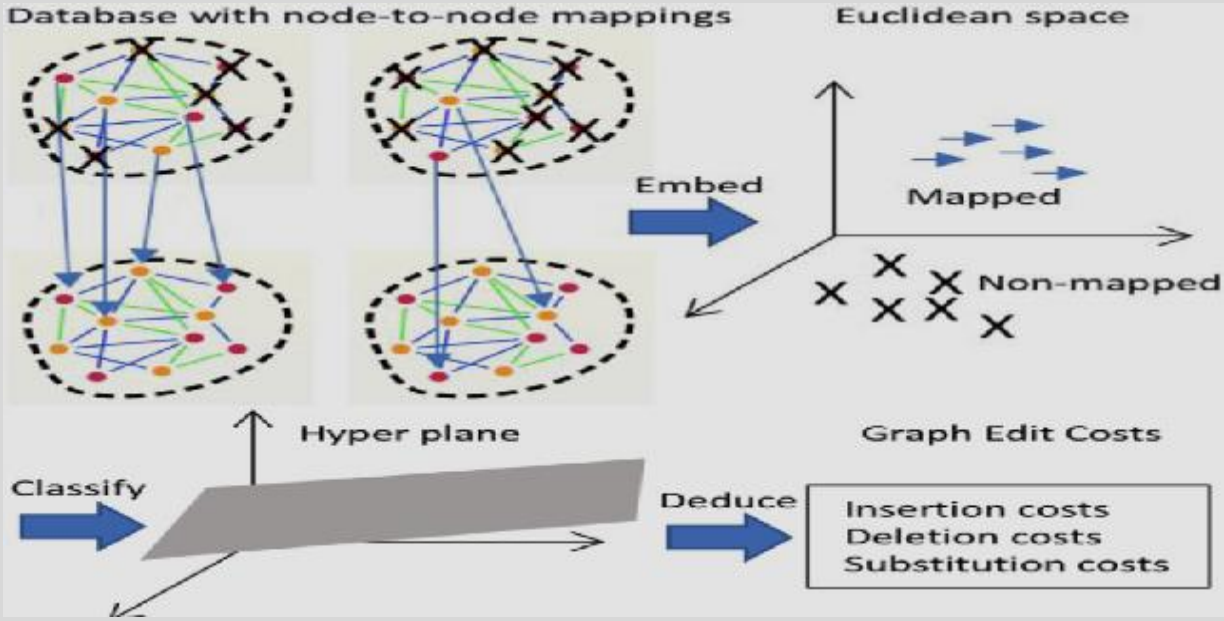
LEARNING THE GRAPH EDIT DISTANCE THROUGH EMBEDDING THE GRAPH MATCHING

Shaima Ahmed Algabli

UNIVERSITAT ROVIRA i VIRGILI

Learning the Graph Edit Distance through Embedding the Graph Matching

SHAIMA ALGABLI



DOCTORAL THESIS
2020

UNIVERSITAT ROVIRA I VIRGILI
LEARNING THE GRAPH EDIT DISTANCE THROUGH EMBEDDING THE GRAPH MATCHING
Shaima Ahmed Algabli

Learning the Graph Edit Distance through Embedding the Graph Matching

DOCTORAL THESIS

Author:

Shaima AL GABALI

Supervisor:

Dr. Francesc Serratosa

Department of Computer Engineering and Mathematics

Tarragona
2020

UNIVERSITAT ROVIRA I VIRGILI
LEARNING THE GRAPH EDIT DISTANCE THROUGH EMBEDDING THE GRAPH MATCHING
Shaima Ahmed Algabli

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 878l

We STATE that the present international study Thesis entitled: “Learning the Graph Edit Distance through embedding the graph matching”, presented by **Shaima Algabli**, 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 it meets all the requirements in order to qualify and obtain the PhD International Mention.

Tarragona, Spain

February, 2020

Thesis Supervisor:

Dr. Francesc Serratosa

To my husband '**Thamer**', I am speechless to describe your scarify and support, we faced alot of obstacles and catastrophes in our life journey, war, death and frustration but your stand and stability let me stand again and again. To you dedicate this work .Without you I will not be able to complete, you are my friend and a great partner beside marvelous father. You are compensating our Lord to me after losing my parents. Love you and be forever with me, you are my real man.

Agraïments

UNIVERSITAT ROVIRA I VIRGILI
LEARNING THE GRAPH EDIT DISTANCE THROUGH EMBEDDING THE GRAPH MATCHING
Shaima Ahmed Algabli

Acknowledgements

I WOULD LIKE TO EXPRESS MY GRATITUDE TO MY SUPERVISOR DR. **FRANCESC SERRATOSA** FOR GIVING ME THE OPPORTUNITY TO WORK UNDER HIS ACADEMIC SUPERVISION AND FOR HIS DEDICATION, GUIDANCE AND SUPPORT DURING THESE YEARS.

DURING THE LAST YEARS, I HAVE MET MANY PEOPLE FROM WHOM I HAVE LEARNT A LOT. TO ALL OF THEM, THANK YOU VERY MUCH FOR EACH OF THE GREAT MOMENTS THAT WE HAVE SHARED AND ALSO FOR THE GREAT TALKS THAT I AM COMPLETELY SURE WE HAVE HAD.

I WOULD ALSO LIKE TO THANK MY HUSBAND **THAMER**, **PARENTS** IN PARADISE, **BROTHERS SISTERS**, AND MY LITTLE BABIES **SAJA** AND **JANA** FOR THEIR LOVE, SUPPORT AND ENCOURAGEMENT. WITHOUT THEM, THIS THESIS WOULD NOT HAVE BEEN POSSIBLE.

LAST BUT NOT THE LEAST, TO MY BLOODED COUNTRY '**YEMEN**' I WANT TO EXPRESS MY HOPE THAT PEACE AND GOOD WILL PREVAIL AND WE WILL RETURN TO IT SOON.

Abstract

Graphs are abstract data structures used to model real problems with two basic entities: nodes and edges. Each node or vertex represents a relevant point of interest of a problem, and each edge represents the relationship between these points. Nodes and edges could be attributed to increase the accuracy of the model, which means that these attributes could vary from feature vectors to description labels. Due to this versatility, many applications have been found in fields such as computer vision, bio-medics, network analysis, and so on.

During more than 30 years, researchers have been focused on how to represent objects through graphs and how to compute the distance between them. The definition of an adequate model to measure the dissimilarity between these representations is a key issue in pattern recognition, which is usually called Error-Tolerant Graph Matching. One of the most popular approaches in order to find a solution to this problem is the Graph Edit Distance, which estimates the distance between a pair of graphs computing the sum of costs of different edit operations that transform one graph into another.

The first part of this thesis presents a general method to automatically learn the edit costs involved in the Graph Edit Distance. The method is based on embedding pairs of graphs and their ground-truth node-to-node mapping into a Euclidean space. In this way, the learning algorithm does not need to compute any Error-Tolerant Graph Matching,

which is the main drawback of other methods due to its intrinsic exponential computational complexity. Nevertheless, the learning method has the main restriction that edit costs have to be constant. Then, we test this method with several graph databases and also we apply it to perform image registration.

In the second part of the thesis, this method is particularized to fingerprint verification. The two main differences with respect to the other method are that we only define the substitution edit costs on the nodes. Thus, we assume graphs do not have edges attributes. And also, the learning method is not based on a linear classification but on a linear regression.

Contents

List of Figures

List of Tables

List of Abbreviations

C Function that represents the cost

e Edge

v Node

f Bijective matching between nodes

$\hat{f}$ Bijective ground truth correspondence

G Graph

s Computational cost of computing the distance between local structures

S the Star of a node

T Set of bijection functions

v Node or vertex

D Number of deleted nodes or edges

Δ Attributes values

γ Function to assign a set of values

γ_v a function to map nodes to their attributed values

γ_e Maps the structure of the nodes

$\mathrm{M_{a,i}}$ The number of substituted external nodes

K_e The cost of assigning edge

$\widetilde{\Sigma_e^p} \sqsubseteq \sum_e^p$ Define the non-existent or null edges

$C_{w_v}\left(v_a^p, v_i^q\right)$ Substitution of the central nodes

$T_{a,i} \cdot (K_v + K_e)$ Deleting or inserting the external nodes

$\check{C}\left(v_a^p, v_i^q\right)$. $\mathrm{T_{a,i}}$Substitution of the external edges and nodes

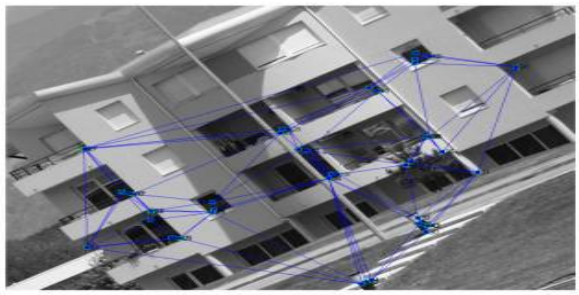
$\left|\mathrm{n}_{\mathrm{N}_\mathrm{a}^\mathrm{p}} - \mathrm{n}_{\mathrm{N}_\mathrm{i}^\mathrm{q}}\right|$ The difference between the orders

UNIVERSITAT ROVIRA I VIRGILI
LEARNING THE GRAPH EDIT DISTANCE THROUGH EMBEDDING THE GRAPH MATCHING
Shaima Ahmed Algabli

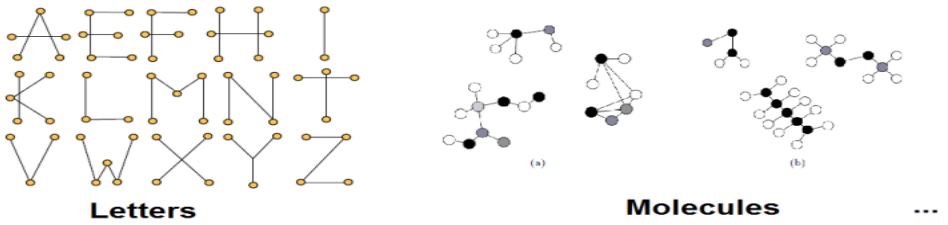
Chapter 1: Introduction

Graphs are a powerful and popular representation formalism in pattern recognition. In a graph-based representation, vertices and their attributes describe objects (or part of objects) while edges represent interrelationships between the objects. Graphs are a really flexible and powerful way to represent data. Due to the improvement of computer capacities, graph have become more and more popular in the field of Pattern Recognition (PR). PR problems can take advantage of a graph in two ways, through Graph Matching, and Graph Embedding. One of the key advantages of using graph structures is that the same representational model is able to fit a wide range of problems from image understanding to interaction networks. Graph data has been applied to machine learning [1], cheminformatics [2], bioinformatics [3], data mining [4], and many others. More precisely, in pattern recognition and computer vision, attributed graphs have been used to represent structural objects that have to be identified or classified. For instance, graphs are successfully applied in representing 2D or 3D objects, handwritten characters, networks, proteins, fingerprints, and so on [5]. Figure 1.1 shows example of graph representation structure. Note that, to generate graphs, the pattern recognition process has to extract them from raw data. This is not a trivial task, since the quality of graph-based representation is crucial for the rest of the process. This thesis does not consider the graph generation process.

(1)



Images



(2)

Fig 1. 1 (1,2) Attributed graphs represent the structural relations

If elements in pattern recognition are modelled through attributed graphs, error-tolerant graph-matching algorithms are needed that aim to compute a correspondence between nodes of two attributed graphs that minimizes some kind of objective function. To that aim, one of the most widely used methods to evaluate an error-correcting graph matching is graph edit distance (GED) [6, 7, 8, 9, 10].

GED is defined as the minimum amount of required distortion to transform one graph into another. To this end, a number of distortion or edit functions consisting of deletion, insertion, and substitution of nodes and edges are defined. To quantitatively evaluate the graph transformation, an edit cost is assigned to each edit operation according to the amount of distortion that it introduces in the transformation.

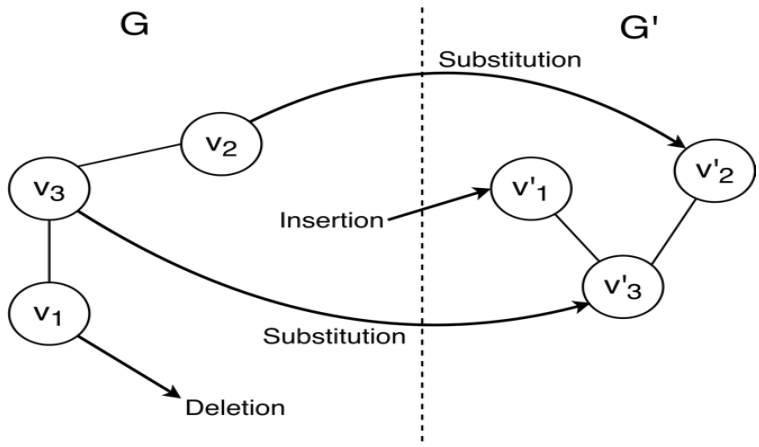


Fig 1. 2: Example of two graphs to be compared. The GED between G and G′ is the cost of substituting two nodes, deleting a node, inserting a node, substituting an edge, deleting an edge and inserting an edge.

For instance, in figure 1.2, GED (G,G′) is defined such that G is transformed into G′ given the transformation highlighted by the arrows. This is done by substituting v3 by v3′ and v2 by v2′. Moreover, v1 is deleted and v1′ is inserted. The distance is the sum of costs of these operations. That is, SubsNode(v3, v3′)+ SubsNode(v2, v2′) + DelNode(v1) + InsNode(v3′) + SubsEdge(v3, v2, v3′ , v2′) + DelEdge(v3, v1)+InsEdge(v3′ , v1′). Other transformations generate other costs and so, the distance becomes the minimum of the costs of all transformations. Thus, the structural and semantic dissimilarity of graphs is only correctly reflected by the GED if the underlying edit costs are defined appropriately and depending on the application at hand.

The thesis breaks down into 5 chapters:

Chapter 1 introduces this thesis explaining the main goals of it.

Chapter 2 is devoted to describing the required concepts for the rest of the document. Concepts described overview of the basics of graph matching, graph edit distance, fingerprint matching and summary of most popular methods have been used in edit cost learning.

Chapter 3 presents a learning method to automatically deduce the insertion and deletion costs of the Graph edit distance. The method is based on embedding the ground-truth node-to-node mappings into Euclidean space and learning the edit costs through the hyper-plane that splits the nodes into mapped ones and non-mapped ones in this new space. The method is applied to image registration.

Chapter 4 presents a learning method applied to fingerprint matching. It is similar to the one presented in the previous chapter but edges are not considered and the parameters are learned based on a regression method instead of a classification method.

Finally, Chapter 5 draws conclusions and gives direction to further work.

Chapter 2: Related Work

This chapter explains the state-of-the-art related to learning graph edit distance. We start by reviewing the approaches used and then, we summarize the existing algorithms that have been using, we review the most related works to the target identification.
This chapter is a review of the literature on graph matching and learning graph edit distance and their definition. All the references commented correspond to problems and methods, although some of them do not correspond with the specific types of problems addressed in this dissertation and are included for the interested reader so that one can have an idea on the different subjects and groups

2.1 Error-Tolerant Graph Matching

Once the problem is modelling with graphs, the main challenge is to classify or compare them. Since early seventies until now, several graph matching algorithms have been released and many surveys have been published [5,11,12]. This challenge inspired us to learn the cost automatically which is needed in graph matching.

The definition of an adequate dissimilarity model between two patterns is one of the most basic requirements in pattern recognition. The process of finding a distance value and a correspondence between two attributed graphs is commonly referred to as Error-Tolerant Graph Matching. There are several

Error-Tolerant Graph Matching models available. Spectral methods, for instance, constitute an important class of Error-Tolerant Graph Matching procedures with a quite long Tradition [11,12,13,14,15].

Graph kernels constitute another important family of similarity measure procedures and various types of graph kernels emerged during the last decade [16,17,18,19,20] For an extensive review on these. Other Error-Tolerant Graph Matching methods developed during the last fifty years, the reader is referred to [21,12,5], but probably the most well-known model for Error-Tolerant Graph Matching is the Graph Edit Distance. Figure 2.1 shows a representation of edit operations: delete edge, delete node, insert node, insert edge and substitute node.

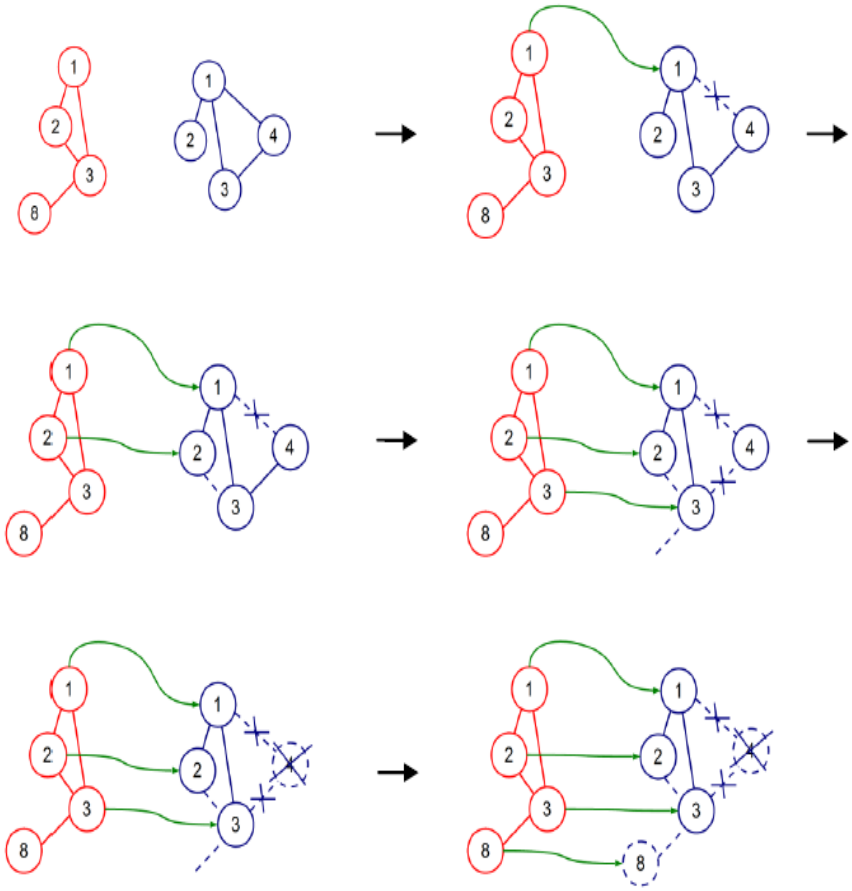


Fig 2. 1: An example of an edit path between two graphs. Green arrows: substitutions. Studded: deletions. Dotted: insertions.

The basic idea behind the graph edit distance (GED) is to define a dissimilarity measure between two graphs, which is defined as the minimum amount of required distortion to transform one graph into the other. To this end, a number of distortion or edit operations, consisting of insertion, deletion and substitution of both nodes and edges are defined. Then, for every pair of graphs (G^p and G^q), there is a sequence of edit operations, or an edit path $editPath(G^p, G^q) = (\sigma_1, \dots, \sigma_k)$ (where each σ_i denotes an edit operation) that transforms one graph into the other. In general, several edit paths may exist between two given graphs. Figure 2.2 shows the edit path that transforms G^p into G^q. It is composed of the following five edit operations: delete edge, delete node, insert node, insert edge and substitute node.

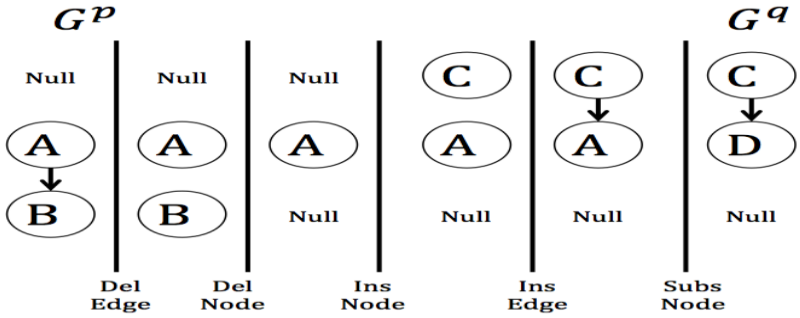


Fig 2. 2: One of the edit paths that transforms G^p into G^q

To quantitatively evaluate which edit path is the best, edit cost functions are introduced. The basic idea is to assign a penalty of cost to each edit operation according to the amount of distortion that it introduces in the transformation.

Each $editPath$ can be related to a correspondence $f^{p,q} \in F$ between the involved graphs (contrarily, given a matching, several edit path could be defined). In this way, each edit operation assigns a node of the first graph to a node of the second graph. Deletion operation (insertion operation) is transformed in to an assignation of a non-null node (null node) of the first graph to a null node (non-null node) of the second graph. Substitutions simply indicate node-to-node assignations.

We define a correspondence between two graphs or simply a correspondence, as follows: Let G^p=$(\sum_v^p, \sum_e^p, \gamma_v^p, \gamma_e^p)$ and G^q=$(\sum_v^q, \sum_e^q, \gamma_v^q, \gamma_e^q)$ be two attributed graphs of initial order and to allow maximum flexibility in the matching process, graphs are extended with null nodes to be of order $n + m$. We refer to null nodes of G^p and G^q by $\widetilde{\Sigma_v^p} \sqsubseteq \sum_v^p and \widetilde{\Sigma_v^q} \sum_v^q$ respectively. Let F be a set of all possible bijections between the two node sets $\Sigma_{\mathrm{v}}^{\mathrm{p}}$ and $\Sigma_{\mathrm{v}}^{\mathrm{q}}$. We define the non-existent or null edges as $\hat{\Sigma}_e^p \subseteq \Sigma_e^p$ and$\hat{\Sigma}_e^q \subseteq \Sigma_e^q$ respectively. Correspondence $f^{p,q}: \Sigma_v^p \to \Sigma_v^q$ is bijective and assigns one node of G^{p} to only one node of G^{q}.

Correspondence$f^{p\to q} : \sum_v^p \to \sum_{v,}^q$, Assigns bijectively one node of to only one node of the correspondence between edges is implicitly defined accordingly to the correspondence of their

terminal nodes. Using this transformation, given two graphs, G^p and G^q, and a bijective matching between their nodes,$f^{p,q}$, the graph edit cost is:

$$EditCost_{K_v,K_e}(G^p, G^q, f^{p,q}) =$$

$$EditCost(G^p, G^q, f)^{p,q} =$$

$$\sum_{\forall\, v_a^p \in \Sigma_v^p - \widehat{\Sigma}_v^p \;s.t.\; v_i^q \in \Sigma_v^q - \widehat{\Sigma}_v^q} C_{w_v}\left(v_a^p, v_i^q\right) + \tag{2.1}$$

$$\sum_{\forall\, e_{ab}^p \in \Sigma_e^p - \widehat{\Sigma}_e^p \;s.t.\; e_{ij}^q \in \widehat{\Sigma}_e^q - \widehat{\Sigma}_e^q} C_{w_e}\left(e_{ab}^p, e_{ij}^q\right) +$$

$$\sum_{\forall\, v_a^p \in \widehat{\Sigma}_v^p \;\; s.t.\; v_i^q \in \Sigma_v^q - \widehat{\Sigma}_v^q} K_v \; +$$

$$\sum_{\forall\, e_{ab}^p \in \widehat{\Sigma}_e^p \;\; s.t.\; e_{ij}^q \in \Sigma_e^q - \widehat{\Sigma}_e^q} K_e \; +$$

$$\sum_{\forall\, v_a^p \in \Sigma_v^p - \widehat{\Sigma}_v^p \;s.t.\; v_i^q \in \widehat{\Sigma}_v^q} K_v \; +$$

$$\sum_{\forall\, e_{ab}^p \in \Sigma_e^p - \widehat{\Sigma}_e^p \;s.t.\; e_{ij}^q \in \widehat{\Sigma}_e^q} K_e$$

If we consider $f^{p,q}\left(v_a^p\right) = v_i^q$ and $f^{p,q}\left(v_b^p\right) = v_j^q$, which forces e_{ab}^p to be mapped to e_{ij}^q. Constant K_v is the cost of deleting node v_a^p of G^p or inserting node v_i^q of G^q. Likewise for the edges, K_e is the

cost of assigning edge e_{ab}^{p} of G^{p} to a non-existing edge of G^{q} or assigning edge e_{ab}^{q} of G^{q} to a non-existing edge of G^{p}. The cases in which two null nodes or null edges are mapped is zero by definition. C_{Wv} is a function that represents the cost of substituting node v_a^p of G^p by node $f^{p,q}(v_a^p)$ of G^q. The same thing occurs with C_{We} but considering edges instead of nodes. These functions have a weight on each attribute represented by $w_v = [w_{v(1)},,,w_{v(N)}]$ in the nodes and $w_e = [w_{e(1)},,,w_{e(M)}]$ in the edges. Then, C_{Wv} and C_{We} are represented as linear combinations of the vector of weights and the vector of local costs:

$$C_{w_v}(v_a^p, v_i^q) = \sum_{t=1}^{N} w_{v(t)} \cdot C_{v(t)}(v_{a(t)}^p, v_{i(t)}^q) \tag{2.2}$$

$$C_{w_e}(e_{ab}^p, e_{ij}^q) = \sum_{t=1}^{M} w_{e(t)} \cdot C_{e(t)}\left(e_{ab(t)}^p, \boldsymbol{e}_{ij(t)}^q\right)$$

Moreover, it is usually considered that the weights are normalised, $|w_v| = 1$ and $|w_e| = 1$. In this case, $w_{v(1)}$ and $w_{e(1)}$ are imposed,

$$w_{v(1)} = 1 - \left|[w_{v(2)},,,w_{v(N)}]\right| \tag{2.3}$$
$$w_{e(1)} = 1 - \left|[w_{e(2)},,,w_{e(M)}]\right|$$

The GED is the minimum cost under any bijection in F:

$$GED_{w_v,w_e,K_v,K_e}(G^p, G^q) = \min_{f^{p,q} \in F} \{EditCost_{w_v,w_e,K_v,K_e}(G^p, G^q, f^{p,q})\} \quad (2.4)$$

Moreover, we define $\dot{f}^{p,q}$ as the correspondence in F such that $EditCost_{w_v,w_e,K_v,K_e}(G^p, G^q, \dot{f}^{p,q})$ is the minimum one.

Figure 2.3 shows an initial pair of graphs G^p and G^q to be corresponded have three nodes each.

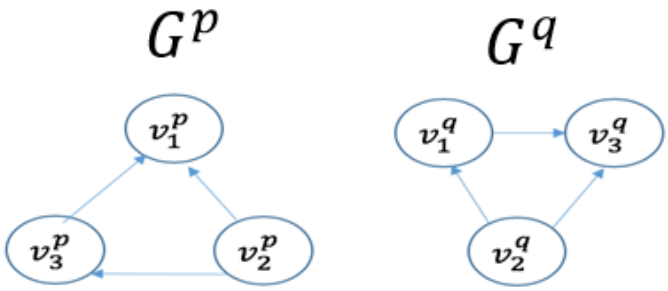


Fig 2. 3: Graphs G^p and G^q.

Figure 2.4 shows the optimal correspondence $\dot{f}^{p,q}$ between them. There are two node substitutions, one node insertion and one node deletion. For this reason, both graphs have been enlarged with one null node.

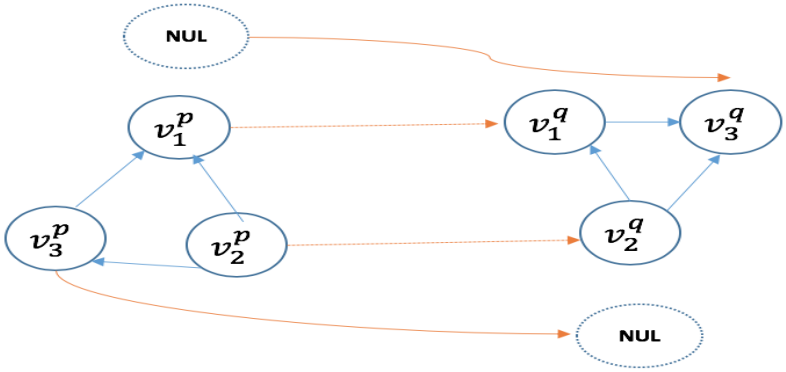


Fig 2. 4: Optimal correspondence $\dot{f}^{p,q}$ between graphs G^p and G^q.

2.2 Computing the Graph Edit Distance

Computing the GED is cast as a shortest path problem, often implemented as an A* search algorithm [24, 25]. These types of problems are considered to be NP-complete [26]. Thus, the computational complexity of these methods is exponential in the number of nodes of the involved graphs. For this reason, greedy algorithms have been presented that deduce a sub-optimal distance and correspondence between nodes in polynomial time. The main idea is to optimise local criteria instead of global criteria. For instance, the Graduated assignment [27], the Bipartite graph matching, [40,29,30-32] or the Greedy edit distance algorithm [33,34]. Table 2.1 shows the computing cost for most popular methods.

References	Method	Computing cost
[24,25]	A*Search	Np complete(exponential)
[33,34]	Greedy algorithm	Polynomial Time
[27]	Graduated assignment	Polynomial Time
[40,29,30-32]	Bipartite graph matching	Polynomial Time

Table 2. 1: Computing cost for most popular methods.

Other algorithms seem to obtain more accurate correspondences at the expense of increasing the computational cost [35] or [36].

These greedy algorithms define a bi-dimensional matrix in which the number of rows or columns is related to the graph order and each cell represents the cost of mapping a pair of nodes (one per graph) and their local structures (for instance, the connected edges and the nodes connected to these edges). Exceptionally, the Believe matching algorithm [37] deduces a node-to-node mapping in linear computational cost and without the computation of the bi-dimensional cost matrix, at the expense of needing an initial small set of node-to-node mappings.

All of these algorithms have the substitution, insertion and deletion costs of nodes and edges as input parameters. To this end, the edit cost between two graphs given a specific node-to-

node mapping f between nodes of both graphs is defined as the addition of the substitution, deletion and insertion costs of their local structures.

The optimal computation of the GED is usually carried out by means of a tree search algorithm, which explores the space of all possible correspondence from nodes and edges of the first graph to the nodes and edges of the second graph. A widely-used method is based on the A* algorithm [28,29] Unfortunately, the computational complexity of this algorithm is exponential in the number of nodes of the involved graphs and for this reason heuristic functions can be used to speed up the exploration of the search space. This means that the runtime may be non-admissible in some applications even for reasonably small graphs.

To this end, the Edit Cost between two graphs (Equation 2.1) is redefined as an addition of the costs of correspondence their Stars. Stars are graph with the structure composed by a node and its adjacent edges and nodes

$$EditCost^{sub}_{w_v,w_e,K_v,K_e}(G^p, G^q, f^{p,q}) = \sum_{\forall\, v_a \in \Sigma_v - \widehat{\Sigma}_v \;\; s.t.\; v_i^{\mathrm{p}} \in \Sigma_v^{\mathrm{p}} - \widehat{\Sigma}_v'} C_{a,i} + \sum_{\forall\, v_a \in \widehat{\Sigma}_v} \sum_{s.t.\; v_i^{\mathrm{p}} \in \Sigma_v^{\mathrm{p}} - \widehat{\Sigma}_v'} C_{a,\varepsilon} + \sum_{\forall\, v_a \in \Sigma_v - \widehat{\Sigma}_v} \sum_{s.t.\; v_i^{\mathrm{p}} \in \widehat{\Sigma}_v'} C_{\varepsilon,i} \tag{2.5}$$

Where $f^{p,q}(v_a^p) = v_i^q$. Besides, $C_{a,i}$ denotes the cost of substituting star S_a^p by star S_i^q. $C_{a,\varepsilon}$ denotes the cost of deleting star S_a^p and $C_{\varepsilon,i}$ denotes the cost of inserting star v_i^q.Similarly to the optimal GED, we define an approximation of the edit distance as the minimum of the edit cost:

$$GED^{sub}_{w_v,w_e,K_v,K_e}(G^p, G^q) = \min_{f^{p,q}\in F}\left\{EditCost^{sub}_{w_v,w_e,K_v,K_e}(G^p, G^q, f^{p,q})\right\} \quad (2.6)$$

And also, we define $\dot{f}_{sub}^{p,q}$ as the correspondence in F such that $EditCost^{sub}_{w_v,w_e,K_v,K_e}(G^p, G^q, \dot{f}_{sub}^{p,q})$ is the minimum one.

Figure 2.5 shows the same two graphs but displayed considering the sub-optimal correspondence$\dot{f}_{sub}^{p,q}$. In this case, both graphs are seen as a set of stars and we have considered $\dot{f}^{p,q}$ and $\dot{f}_{sub}^{p,q}$ to be the same.

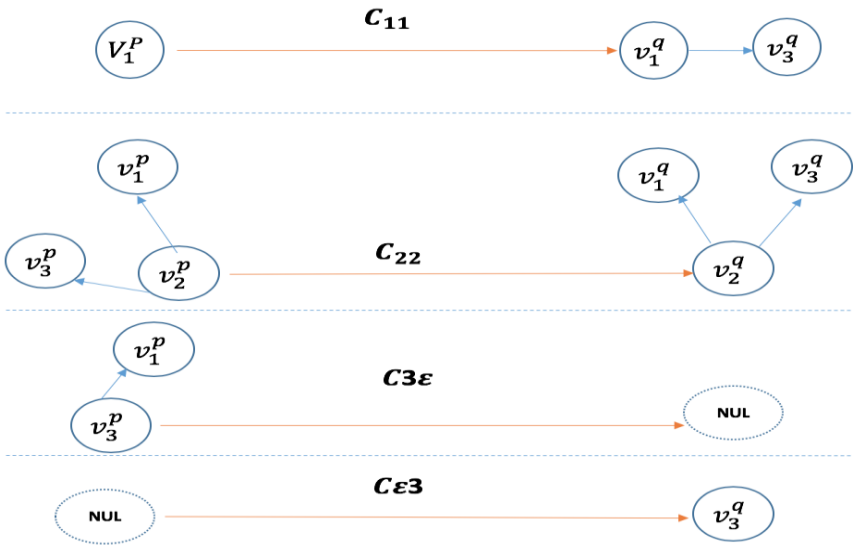


Fig 2. 5: Suboptimal correspondence $\dot{f}_{sub}^{p,q}$ between graphs G^p and G^q

Note that algorithms [30,31] and [40], which compute the sub-optimal GED (Equation 2.6), assume that correspondence $\dot{f}_{sub}^{p,q}$ tends to be similar to $\dot{f}^{p,q}$ (deduced through Equation 2.4) although the computational cost of computing (Equation 2.6) is polynomial but the computational cost of computing (Equation 2.4) is exponential. Our method is based on the same assumption. Thus, we believe that the parameters learned in our methodology tend to be the same as the parameters learned in an optimal way, which would be almost impossible to deduce due to its runtime.

The type of local structure used by the sub-optimal graph matching algorithms was analysed in depth in [32].

The conclusion of that thesis was that the star is the local structure that achieves the best relation between optimality and computational cost. The costs of substituting, deleting and inserting stars are defined as the GED between two stars (Equations 2.1 and 2.4). In the first case, both stars have a non-empty structure. In the other cases, one of the stars has an empty structure:

Substitution: $C_{a,i} = C_{w_v}\left(v_a^p, v_i^q\right) + T_{a,i} \cdot (K_v + K_e) + \check{C}\left(v_a^p, v_i^q\right)$
Deletion: $C_{a,\varepsilon} = K_v + n_{N_a^p} \cdot (K_v + K_e)$
Insertion: $C_{\varepsilon,i} = K_v + n_{N_i^q} \cdot (K_v + K_e)$ (2.7)

The star substitution cost is composed of three terms: substitution of the central nodes: $C_{w_v}\left(v_a^p, v_i^q\right)$, deleting or inserting the external nodes: $T_{a,i} \cdot (K_v + K_e)$ and substitution of the external edges and nodes: $\check{C}\left(v_a^p, v_i^q\right)$. $\mathrm{T_{a,i}} = \left|\mathrm{n_{N_a^p}} - \mathrm{n_{N_i^q}}\right|$ is the difference between the order of stars. Function $\check{C}\left(v_a^p, v_i^q\right)$ is computed through a linear solver [77] that obtains the best correspondence g between external nodes of the star. Therefore,

$$\check{C}\left(v_a^p, v_i^q\right) = \sum_{a'=1}^{M_{a,i}} \left[C_{w_v}\left(v^p_{\{a'\}}, v^q_{g(\{a'\})}\right) + C_{w_e}\left(e^p_{a\{a'\}}, e^q_{i\,g(\{a'\})}\right)\right] \quad (2.8)$$

Where $M_{a,i}$ is the number of substituted external nodes, $M_{a,i} = \min\left\{n_{N_a^p}, n_{N_i^q}\right\}$. Node $v_{\{a'\}}^p$ represents the a′th external node of the star.

Bipartite algorithm (BP) [40,39] is one of the newest methods presented to solve the GED and new optimisation techniques of this algorithm have recently appeared [41,42] experimental validation shows that, currently it is one of the best sub-optimal algorithms since it obtains a good approximation of the distance value in cubic computational cost. BP algorithm is composed of three main steps. The first step defines a cost matrix, the second step applies a lineal solver such as the Hungarian method [43] or the Jonker-Volgenant method [75] to this matrix and deduces the matching $f^{p,q}$.

The third step computes the edit cost given this matching between both graphs $EditCost\left(G^p, G^q, f^{p,q}\right)$.

2.3 Fingerprint Matching

A fingerprint is an impression left by the friction ridges of a human finger. They are easily deposited on suitable surfaces, such as glass or metal, by the natural secretions of sweat from the eccrine glands that are present in epidermal ridges. Moreover, deliberate impressions of fingerprints may be formed by ink transferred from the peaks of friction ridges on the skin to a relatively smooth surface. Fingerprints are considered to be nearly unique, difficult to alter, and durable over the life of an individual, making them suitable as long-term markers of human identity. Currently, thousands of applications have been deployed that identify humans through digital fingerprint scanners [78].

Fingerprints are usually represented by a set of minutiae. In an identification process, it is needed a distance between a pair of sets of minutiae to deduce how similar the original fingerprints are [79]. Moreover, this distance has some parameters that have to be tuned to achieve a good accuracy. Usually, these parameters are manually tuned or in a pseudo automatic manner, based on the accuracy obtained in the learning set. In the present thesis, we present a novel semi-automatic learning algorithm to deduce the distance parameters.

A minutia, in fingerprint identification, refers to the various ways in which the skin ridges can be discontinuous. For example, a ridge can abruptly come to an end, then we have a termination, or can divide into two ridges, then we have a bifurcation. Although other types of minutiae can be considered, usually only a coarse classification into these two types is adopted to deal with the practical difficulty in automatically discerning the different types with high accuracy. Moreover, we also have the information of the position were the end of the ridge has arrived and also the local ridge orientation at this position, crossing through an arbitrary small neighbourhood centred at this position. Thus, the ath minutia m^a in a fingerprint is represented as a 3-tuple composed of the position in the image: $m^a_{xy} \in [1 \dots width, 1 \dots \text{height}]$, the angle: $m^a_{\alpha} \in [0 \dots 2\pi]$ and the type of minutia: $m^a_{TB} \in \{Terminal, Bifurcation\}$,

$$m^a = \{m^a_{xy}, m^a_{\alpha}, m^a_{TB}\} \tag{2.9}$$

In a classification process, it is needed to define a distance between the elements at hand. In our case, before defining a distance between fingerprints, we need to define a minutia distance. A usual option [62] is to define the minutia distance, MD, as a linear combination of the distances of the three minutia terms as follows,

$$MD_{w_{xy},w_\alpha}(m^a, m^i) = w_{xy} \cdot d_{xy}(m^a_{xy}, m^i_{xy}) + w_\alpha \cdot d_\alpha(m^a_\alpha, m^i_\alpha) + d_{TB}(m^a_{TB}, m^i_{TB}) \quad (2.10)$$

Subject to

$$w_{xy}{}^2 + w_\alpha{}^2 = 1 \quad (2.11)$$

d_{xy} refers to the distance between the minutiae's locations and it is defined through the Euclidean distance. d_α is the distance between the minutiae's angles. It is defined as $d_\alpha(m^a_\alpha, m^i_\alpha) = min(|m^a_\alpha - m^i_\alpha|, 2\pi - |m^a_\alpha - m^i_\alpha|)$ to take into consideration that the angles are cyclic. Finally, d_{TB} is the distance between the minutia's type, terminal or bifurcation. Usually, the correspondence between different types of minutiae is banned, then we have:

$$d_{TB}(m^a_{TB}, m^i_{TB}) = \begin{cases} 0 & if\ m^a_{TB} = m^i_{TB} \\ \infty & otherwise \end{cases} \quad (2.12)$$

Note that the weights w_{xy} and w_α tune how much important is each element in the triplet and these are the weights we learn through our method. Note there is not weight on d_{TB} in Equation 2.10 since this distance becomes null or infinity (Equation 2.12).

A fingerprint F is composed of a set of minutiae, $F = \{m^1, \dots, m^{|F|}\} Being\ |F^p|$ the number of minutia of F. Having defined the distance between minutiae, we can define the fingerprint distance FD. This distance is usually computed as the minimum value given all possible correspondences between minutiae of both fingerprints. The distance between fingerprints F^{p} and F^{a} is defined as follows,

$$FD_{w_{xy},w_\alpha}(F^p, F^a) = \min_{\forall\, f \in \Omega} \left\{ \frac{1}{|F^p|} \sum_{\forall\, m^a \in F^p} MD_{w_{xy},w_\alpha} \left(m^a, f(m^a) \right) \right\} \quad (2.13)$$

We assume, for theoretical reasons, that the number of minutiae of F^{p} is equal or lower than the number of minutiae of F^a, $|F^{\mathrm{p}}| \leq |F^{\mathrm{a}}|$. If it was not the case, we could swap the fingerprints. Thus, we can define the set Ω as a set of injective functions from the fingerprint F^{p} to the fingerprint F^a.Moreover, we define $\dot{f}^{p,\mathrm{a}}$ as the correspondence in Ω that obtains the minimum value.

$$\dot{f}^{p,a} = \underset{\forall\, f \in \Omega}{argmin} \left\{ \frac{1}{|F^p|} \sum_{\forall\, m^a \in F^p} MD_{w_{xy},w_\alpha} \left(m^a, f(m^a) \right) \right\} \quad (2.14)$$

Several algorithms have been presented to compute Equation 2.13 [43] or [75]. Which are out of the scope of this thesis. To reduce the computational cost, these algorithms, by construction, do not allow mapping minutia of different types, which generates an

infinity distance (Equation 2.12). For this reason, the correspondence that generate an infinity value are discarded from Ω.

2.4 Learning the Edit Cost Methods

We realised that in the process of learning the costs, there are two types of optimisation functions. The ones related on the recognition ratio given a classified test set and the ones related to the Hamming distance between a ground truth matching and the automatically deduced matching. Hamming distance is error measure of structural similarity between deduced labelling and ground truth. One of the drawbacks of the first type of functions is that graphs in the dataset need to be classified. This is not the case with the second function but a ground truth matching is needed, therefore, the learning elements in the dataset must be composed of two graphs and a ground truth matching between nodes. To that aim, a new publically available repository of databases of graphs was presented [56]. In this repository, registers in the databases are not conformed to the classical structure of a graph and its class. Contrarily, they are composed of a pair of graphs, their class and also a ground-truth matching between them.

Another important feature of these methods is the type of costs in the learning algorithm obtains such as method obtains in a self-organising map and the method deduces a probability density

function. Therefore, in these cases, classical graph matching algorithms have to be adapted to be applied to these learning methods since these matching algorithms assume edit costs are real numbers. Methods solve problems closer to our method. They assume nodes and edges have several attributes (for instance features obtained by SIFT descriptors) and these methods obtain a weight for each individual feature. In these cases, there is not a unique substitution cost on nodes and on edges but a vector of substitution weights, one for each individual feature. Moreover, insertion and deletion costs are not learned.

There are some graph databases that nodes and edges have only one or any attributes. In these cases, it makes no sense to learn the substitution weights for each feature as in, but it is crucial to learn the best combinations of insertion and deletion costs on nodes and edges as unique real numbers.

In the next paragraph, the presented methods to learn the GED are summarized. In these paragraphs $F_{vs}, F_{es}, F_{vd}, F_{ed}, F_{vi}, F_{ei}$ are the learned edit costs $C_{vs}, C_{es}, C_{vd}, C_{ed}, C_{vi}, C_{ei}$ when they are represented as functions, for instance, the output of a neural network. Moreover, $K_{vd}, K_{vi}, K_{ed}, K_{ei}$ are the learned edit costs $C_{vd}, C_{vi}, C_{ed}, C_{ei}$ when they are represented as constants (a Real number). Finally, $w_{vs} = (w_{vs(1)}, \dots, w_{vs(K_v)})$ and $w_{es} = (w_{es(1)}, \dots, w_{es(K_e)})$ are the weights of the

weighted Euclidean distance when C_{vs} and C_{es} are defined as a weighted Euclidean distance.

Michel Neuhaus and **Horst Bunke**, [46] in 2002 present a technique based on a Self-Organizing Map (SOM). The purpose of this technique is to show how to issue learning graph edit distance cost functions numerically labelled graphs from a corpus of sample graphs. Representing attribute distance spaces that encode edit operation costs. The SOM, are iteratively adapted to minimize the edit distance of those graphs that are required to be similar. To demonstrate the learning effect, the distance model is applied to graphs representing line drawings and diatoms.

The major conclusion to this study was that is an attempt toward such an edit cost learning procedure that is based on SOMs. In order to learn the weights of a SOM to define the substitution, deletion and insertion costs on nodes and edges. These costs become the output of the SOM when the input is the attribute of the node or the edge.

Learns: $F_s^n, F_D^n, F_I^n, F_s^e, F_D^e, F_I^e$

Michel Neuhaus and **Horst Bunke,** [47] in 2007 present an automatic method to learn cost functions from a labelled sample set of graphs. They formulate the graph edit process in a stochastic context and perform a maximum likelihood parameter estimation of the distribution of edit operations. The underlying distortion model is learned using an Expectation Maximization algorithm. The major conclusion to this study is that propose a method to derive graph edit costs from a probabilistic model. Edit costs are used to compute distances between graphs by

performing a structural matching. They introduce a probabilistic model for graph edit operations and show how to estimate the edit operation distribution from a labelled set of graphs. The edit costs are adapted so as to decrease the distance between graphs from the same class, leading to compact graph clusters. The method learns the parameters of a Probability Density Function (PDF) to define the substitution, deletion and insertion costs on nodes and edges. These costs become the inverse of the probability set by the PDF given the attributes of the node or the edge.
Learns: $F_s^n, F_D^n, F_I^n, F_s^e, F_D^e, F_I^e$.

Tiberio S. Caetano, **Li Cheng**, Quoc **V. Le** and **Alex J. Smola**, [48] in 2009 present how to estimate compatibility functions such that the solution of the resulting graph matching problem. Present a method for learning graph matching. The method learns the weights of the weighted Euclidean distance to define the substitution cost on nodes and edges. The substitution cost becomes the weighted Euclidean distance for nodes and edges. Insertion and deletion of nodes and edges are not learned and assumed to be constant.
Learns: w_s^n, w_s^e

Marius Leordeanu, **Rahul Sukthankar**, and **Martial Hebert**, [49] in 2012 show how to perform parameter learning in an unsupervised fashion that is when no correct correspondences between graphs are given during training. The experiments reveal that unsupervised learning compares favourably to the supervised case, both in terms of efficiency and quality, while avoiding the tedious manual labelling of ground truth correspondences. The major conclusion to this study was efficient way of performing both supervised and unsupervised learning for graph matching.
Learns: w_s^n, w_s^e

Xavier Cortés and **Francesc Serratosa**, [50] in 2015 present an interactive method for the image alignment problem based on partially supervised correspondence. Extracted from the images through a Human–Machine Interface and the Interactive Correspondence Method computes a new image alignment and a set of point correspondences.

The major conclusion to this study was to present an interactive method that, given two images taken from different robots, the oracle imposes some correspondences between their local parts to estimate an initial alignment and to restrict the point correspondences.

The method learns the deletion and insertion costs on nodes and edges as constants (Real numbers). The substitution cost is assumed the Euclidean distance between the attributes on nodes or on edges.Learns: $K_D^n, K_I^n, K_D^e, K_I^e$

Xavier Cortes and **Francesc Serratosa**, [51] in 2016 reveal a supervised method to learn the weights that gauges the importance of each node and edge attribute in the distance function between attributed graphs. To do so, the learning algorithm minimizes the difference between the ground truth correspondences of pairs of graphs and their automatically obtained correspondences. The major conclusion to this study show that the distance between the edit cost of the ground truth and the edit cost of the automatically obtained correspondence is a good loss function.

The method learns the weights of the weighted Euclidean distance to define the substitution cost on nodes and edges. The substitution cost becomes the weighted Euclidean distance for nodes and edges. Insertion and deletion of nodes and edges are not learned and assumed to be constant.
Learns: w_s^n, w_s^e.

Xavier Cortés, **Donatello Conte**, **Hubert Cardot**, and **Francesc Serratosa**, [52] in 2018 work in A Deep Neural Network Architecture to Estimate Node Assignment Costs for the Graph Edit Distance. Propose a Model to estimate only the assignments costs minimizing the hamming distance. The model to compute the assignments costs for the Graph Edit Distance by means of a Deep Neural Network previously trained with a set of pairs of graphs properly matched.

The major conclusion to this study to estimate assignment costs for the Graphs Edit Distance using a Deep Neural Network. The method learns the substitution functions on nodes and edges through a Neural Network (NN). The substitution cost is defined as the output of the NN when the input is the attribute on the nodes and edges. Insertion and deletion of nodes and edges are not learned and assumed to be constants.
Learns: F_s^n, F_s^e

Francesc Serratosa and Santacruz, [53] in 2018, present a method in Learning the suboptimal graph edit distance edit costs based on an embedded model, present a general framework to automatically learn these edit costs considering graph edit distance is computed in a sub-optimal way. Then, specify this framework in two different models based on neural networks and probability density functions.

The major conclusion to this study Similar to [52] and [45] but the insertion and deletion costs on nodes and edges are also learned. There is also a NN for insertion and another one for deletion the nodes and edges.
Learns: $F_s^n, F_D^n, F_I^n, F_s^e, F_D^e, F_I^e$

Maxime Martineau, Romain Raveaux, and Donatello Conte, Gilles Venturini, [54] in 2018 propose method in learning error-correcting graph matching with a multiclass neural network in this method an effective scheme for optimizing the graph matching

problem in a classification context. For this, we propose a representation that is based on a parameterized model graph, and optimize the associated parameters. The objective of the optimization problem is to increase the classification rate.

The main conclusion is that this method learns the weights on each node or edge. These weights depend on how important the nodes and edges are to describe the class.

Learns: weights on nodes and edges

Chapter 3: Learning the Graph Edit Distance

3.1 Introduction

We present a learning method to automatically deduce the insertion, deletion and substitution costs of the Graph edit distance. The method is based on embedding the ground-truth node-to-node mappings into a Euclidean space and learning the edit costs through the hyper plane that splits the nodes into mapped ones and non-mapped ones in this new space. In this way, the algorithm does not need to compute any graph matching process, which is the main drawback of other methods due to its intrinsic exponential computational complexity. Nevertheless, our learning method has two main restrictions:

1) The insertion and deletion edit costs have to be constants.

2) The substitution edit costs have to be represented as inner products of two vectors. One vector represents certain weights and the other vector represents the distances between attributes. Experimental validation shows that the matching accuracy of this method outperforms the current methods. Furthermore, there is a significant reduction in the runtime in the learning process.

We want to learn parameters w_v, w_e, K_v and K_e through a supervised learning method. The aim of the method is to deduce these parameters so that the optimal correspondences $\dot{f}^{p,q}$ become close to the ground-truth correspondences $\hat{f}^{p,q}$ for all pairs of graphs (G^p, G^q). These ground truth correspondences have been computed by an external system (Human or artificial) and they are considered to be the best correspondence

for our learning purposes. Note that these ground truth Correspondences are independent of the Graph edit distance parameters: w_v, w_e, K_v and K_e.

As an example of the generation of this ground truth correspondence, we could mention fingerprint matching. Given two fingerprints, a specialist decides which the best correspondence between minutiae of these fingerprints is. Thus, the specialist knows nothing about the Graph edit distance and therefore the correspondence that the specialist decides is not influenced by the Graph edit distance parameters.

Given the star costs in Equation 2.7 in chapter 2 and the ground truth correspondence$\hat{f}^{p,q}$, we discern between three options (Figure 3.1)

- First option. Non-null stars are mapped and so we can represent the node-to-node mapping as $\hat{f}^{p,q}(v_a^p) = v_i^q$ where $\mathrm{v_a^p} \in \Sigma_{\mathrm{v}}^{\mathrm{p}} - \hat{\Sigma}_{\mathrm{v}}^{\mathrm{p}}$ and$\mathrm{v_i^q} \in \Sigma_{\mathrm{v}}^{\mathrm{q}} - \hat{\Sigma}_{\mathrm{v}}^{\mathrm{q}}$. It may hold that mapping these two stars is costless (the star substitution costs is lower compared to deleting star $\mathrm{S_a^p}$ or inserting $\mathrm{S_i^q}$. Thus, it would hold that $\mathrm{C_{a,i}} \leq \mathrm{C_{a,\varepsilon}}$ and also $\mathrm{C_{a,i}} \leq \mathrm{C_{\varepsilon,i}}$ since the selected matching has to be the one with the minimum cost.

- Second option. A star of the first graph is deleted and it is represented as a non-null node to null node mapping $\hat{f}^{p,q}(v_a^p) = v_i^q$, so that $\mathrm{v_a^p} \in \Sigma_{\mathrm{v}}^{\mathrm{p}} - \hat{\Sigma}_{\mathrm{v}}^{\mathrm{p}}$ and $\mathrm{v_i^q} \in \hat{\Sigma}_{\mathrm{v}}^{\mathrm{q}}$. In this case, it may hold that deleting the star of $\mathrm{G^p}$ is costless compared to substituting it with any of the stars in $\mathrm{G^q}$. Then, it may hold that $\mathrm{C_{a,\varepsilon}} \leq \mathrm{C_{a,k}}$ for all $\mathrm{v_k^q} \in \Sigma_{\mathrm{v}}^{\mathrm{q}} - \hat{\Sigma}_{\mathrm{v}}^{\mathrm{q}}$

- Third option. A star on the second graph is inserted and so it is represented as $\hat{f}^{p,q}(v_a^p) = v_i^q$ where $\mathrm{v_a^p} \in \hat{\Sigma}_{\mathrm{v}}^{\mathrm{p}}$ and $\mathrm{v_i^q} \in \Sigma_{\mathrm{v}}^{\mathrm{q}} - \hat{\Sigma}_{\mathrm{v}}^{\mathrm{q}}$. In this

case, it may hold that inserting the star of G^q is costless compared to substituting it with any of the stars of G^p. Then, it may hold that $C_{\varepsilon,i} \leq C_{k,i}$ for all $v_k^p \in \Sigma_v^p - \hat{\Sigma}_v^p$.

Summarising:

First option: Substitution: $C_{a,i} - C_{a,\varepsilon} \leq 0$ & $C_{a,i} - C_{\varepsilon,i} \leq 0$

Second option: Deletion: $C_{a,k} - C_{a,\varepsilon} \geq 0, \forall v_k^q$

Third option: Insertion: $C_{k,i} - C_{\varepsilon,i} \geq 0$,

Considering these restrictions, we propose $C_{a,i} - C_{a,\varepsilon} = 0$ to be a good choice to distinguish mappings of the first option from mappings of the second option and therefore to set the border between these two cases. The same thing occurs with equation $C_{a,i} - C_{\varepsilon,i} = 0$, which can be used to distinguish mappings of the first option from mappings of the third option. Note functions $C_{a,i}$, $C_{a,\varepsilon}$ and $C_{\varepsilon,i}$ represent the cost of substituting, deleting and inserting stars that depend on constants w_v, w_e, K_v and K_e. Thus, these constants play a role in the definition of the borders $C_{a,i} - C_{a,\varepsilon} = 0$ and $C_{a,i} - C_{\varepsilon,i} = 0$, which will be used to learn w_v, w_e, K_v and K_e.

In order to make clear explanatory as possible, from now to the end of this chapter in this thesis, we shall not consider the third option (Insertion) again. This is because the whole equations that relate deletion operations can be easily transformed into insertion operations. Moreover, an insertion in $f^{p,q}$ becomes a deletion in $f^{q,p}$ since they are bijective.

3.2 Embedding the node-to-node mappings

Suppose that $S = (S_{v(1)}, \dots, S_{v(N)}, S_{e(1)}, \dots, S_{e(M)}, S_{k_e})$ is a Euclidean $N + M + 1$ dimensional space. N and M are the number of attributes on nodes and edges, respectively. Suppose the ground truth correspondence $\hat{f}^{p,q}$ imposes $\sim N$ node-to-node substitutions. In this case, the substitution of the node v_a^p by the node v_i^q is represented by a point in S. Moreover, the deletion of the node v_a^p is represented by $\sim N$ points in S.

Our learning method is based on two steps:

First, embedding the ground truth mappings (substitutions and deletions) into this Euclidean space. Each substitution is transformed into a point in the embedding space and the class "+1" is assigned to it. If the node v_a^p is mapped into the node v_i^q, then the position of the point that this mapping generates in the embedding space depends on the cost $C_{a,i}$ of substituting their stars.

Each deletion is transformed into $\sim N$ points in the embedding space and the class "-1" is assigned to them. If the node v_a^p is deleted, then the positions of the $\sim N$ points that this deletion generates in the embedding space depends on the costs $C_{a,j}$ of substituting the star S_a^p into the $\sim N$ stars S_j^q, where $v_j^q \in \Sigma_v^q$ (the nodes in G^q that are substituted). Figure 3.1 shows the embedding of the mapping in Figure 2.4 in chapter 2. We suppose that there is only one attribute on the nodes and edges, $N = 1$ and $M = 1$. Thus, S is a 3D Euclidean space, $S = (S_{v(1)}, S_{e(1)}, S_{k_e})$. We see the

two points generated by the two substitutions (points with class "+1"). We also see the two points generated by the only one deletion (points with class "-1") since G^q has two substituted nodes.

Second, finding the border between points of class "+1" and points of class "-1" for all points in the space S by any linear classifier, assuming it is a hyper plane. This is because the Euclidean space has been defined so that parameters w_v, w_e, K_v and K_e are the offset and slope of this hyper plane as follows, (considering equations in chapter 2)

$$\begin{aligned} &\mathrm{S}_{\mathrm{v}(1)} + \mathrm{w}_{\mathrm{v}(2)} \cdot S_{\mathrm{v}(2)} + w_{\mathrm{v}(3)} \cdot S_{\mathrm{v}(3)} + \cdots + w_{\mathrm{v}(\mathrm{N})} \cdot \mathrm{S}_{\mathrm{v}(\mathrm{N})} + \\ &S_{\mathrm{e}(1)} + w_{\mathrm{e}(2)} \cdot S_{\mathrm{e}(2)} + w_{\mathrm{e}(3)} \cdot S_{\mathrm{e}(3)} + \cdots + \mathrm{w}_{\mathrm{e}(\mathrm{M})} \cdot S_{\mathrm{e}(\mathrm{M})} + \\ &\qquad K_e \cdot \mathrm{S}_{k_e} + K_v = 0 \end{aligned} \tag{3.1}$$

In figure 3.1, we also see the hyper plane $\mathrm{S}_{\mathrm{v}(1)} + S_{\mathrm{e}(1)} + K_e \cdot \mathrm{S}_{k_e} + K_v = 0$, which separates both classes. Therefore, the Graph edit distance parameters K_v and K_e are obtained directly from the hyper plane. Note that in this specific case ($N = 1$ and $M = 1$), it turns out that $\mathrm{w}_{\mathrm{e}(1)} = 1$ and $\mathrm{w}_{\mathrm{v}(1)} = 1$ due to Equation 2.3 in chapter 2. For this reason, these terms do not appear in the definition of this hyper plane.

In the rest of this chapter, we show in detail how a mapping from v_a^p to v_i^q is embedded in the spaceS. In this the Euclidean space S so that the border $C_{a,i} - C_{a,\varepsilon} = 0$ for all points in the space S becomes the hyper plane defined in Equation 3.1.

If we consider Equation 2.7 in chapter 2, the border $C_{a,i} - C_{a,\varepsilon} = 0$ between elements of the first and the second option in the system S is rewritten as follows ($T_{a,i}$ is defined in Equation 2.8 in chapter 2),

$$C_{w_v}\left(v_a^p, v_i^q\right) + T_{a,i} \cdot (\mathrm{K}_v + \mathrm{K}_e) + \check{\mathrm{C}}\left(v_a^p, v_i^q\right) - \left(\mathrm{K}_v + \mathrm{n}_{N_a^p} \cdot (\mathrm{K}_v + \mathrm{K}_e)\right) = 0 \tag{3.2}$$

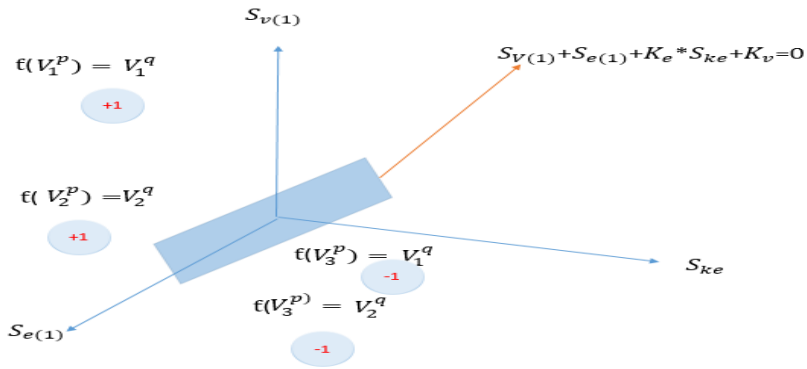


Fig 3. 1: The learning process in the case where there is only one attribute on the nodes and on the edges

Rearranging the terms, we arrive at the expression,

$$\frac{C_{w_v}\left(v_a^p, v_i^q\right) + \check{C}\left(v_a^p, v_i^q\right)}{T_{a,i} - n_{N_a^p} - 1} + K_e \cdot \frac{T_{a,i} - n_{N_a^p}}{T_{a,i} - n_{N_a^p} - 1} + K_v = 0 \tag{3.3}$$

Moreover, the expression $C_{w_v}(v_a^p, v_i^q) + \check{C}(v_a^p, v_i^q)$ in Equation 3.3 can be defined as follows (considering Equation 3.1),

$$\begin{aligned} C_{w_v}(v_a^p, v_i^q) + \check{C}(v_a^p, v_i^q) &= Z_{v\,a,i(1)} + Z_{e\,a,i(1)} + \\ &+ w_{v(2)} \cdot (Z_{v\,a,i(2)} - Z_{v\,a,i(1)}) + , , , + + w_{v(N)} \cdot (Z_{v\,a,i(N)} - Z_{v\,a,i(1)}) + \\ &+ w_{e\,(2)} \cdot (Z_{e\,a,i(2)} - Z_{e\,a,i(1)}) + , , , + \quad + w_{e(M)} \cdot (Z_{e\,a,i(M)} - Z_{e\,a,i(1)}) \end{aligned} \tag{3.4}$$

Note that terms $w_{v(1)}$ and $w_{e(1)}$ do not exist due to Equation 2.3 in chapter 2. Moreover, terms $Z_{v\,a,i(t)}$ and $Z_{e\,a,i(t)}$ are defined as follows,

$$\begin{aligned} Z_{v\,a,i(t)} &= C_{v(1)}(v_{a(t)}^p, v_{i(t)}^q) + \sum_{a'=1}^{n_{N_a^p}} C_{v(t)}\left(v_{\{a'\}(t)}^p, v_{g(\{a'\})(t)}^q\right) \\ Z_{e\,a,i(t)} &= \sum_{a'=1}^{n_{N_a^p}} C_{e(t)}\left(e_{a,\{a'\}(t)}^p, e_{i,g(\{a'\})(t)}^q\right) \end{aligned} \tag{3.5}$$

$v_{\{a'\}}^p$ represents the a'th external node of the star and the term (t) represents the tth attribute of the node or edge. Moreover, g is the mapping between external nodes of the star.

Finally, taking into consideration from Equation 3.3 to Equation 3.6, the node-to-node mapping from v_a^p to v_i^q is embedded in the space S at the point $(S_{v\,a,i(1)}, \dots, S_{v\,a,i(N)}, S_{e\,a,i(1)}, \dots, S_{e\,a,i(M)}, S_{k_e\,a,i})$ as follows,

$$
\begin{aligned}
S_{v\,a,i(t)} &= \frac{Z_{v\,a,i(t)} - Z_{v\,a,i(1)}}{n_{N_a^p}+1-T_{a,i}} && \text{if } t \geq 2 \\
S_{v\,a,i(1)} &= \frac{Z_{v\,a,i(1)}}{n_{N_a^p}+1-T_{a,i}} && \text{otherwise} \qquad (3.6) \\
S_{e\,a,i(t)} &= \frac{Z_{e\,a,i(t)} - Z_{e\,a,i(1)}}{n_{N_a^p}+1-T_{a,i}} && \text{if } t \geq 2 \\
S_{e\,a,i(1)} &= \frac{Z_{e\,a,i(1)}}{n_{N^p}+1-T_{a,i}} && \text{otherwise}
\end{aligned}
$$

In the case that $n_{N_a^p} + 1 - T_{a,i} = 0$, Equation 3.3 and Equation 3.6 take infinity values. These specific cases are not considered, which means that these concrete substitutions are not embedded in the space S.

Considering that $\mathrm{T_{a,i}} = \left|\mathrm{n_{N_a^p}} - \mathrm{n_{N_i^q}}\right|$, these are the cases where $\mathrm{n_{N_a^q}} = 2 \cdot \mathrm{n_{N_i^q}} - 1$. In other words, the number of neighbours of the star in $\mathrm{G^q}$ is almost twice the number of neighbours of the star in $\mathrm{G^p}$.

Note that all the axes are normalized by the structural functional $n_{N_a^p} + 1 - T_{a,i}$. Moreover, axis $S_{k_e\,a,i}$ keeps only the structural information and it is not influences by the attributes. Axes $S_{v\,a,i(t)}$ (for all t) are only related on the node attributes and, similarly, axes $S_{e\,a,i(t)}$ (for all t) are only related on the edge attributes.

Therefore, there is not any axis that relates the attributes on the nodes and on the edges. Expressions $Z_{v\ a,i(t)} - Z_{v\ a,i(1)}$ and $Z_{e\ a,i(t)} - Z_{e\ a,i(1)}$ are the substitution costs on nodes and on edges of the t- attribute subtracted by the substitution costs on nodes and on edges of the first attribute. This is done to force the sum of the weights to be one.

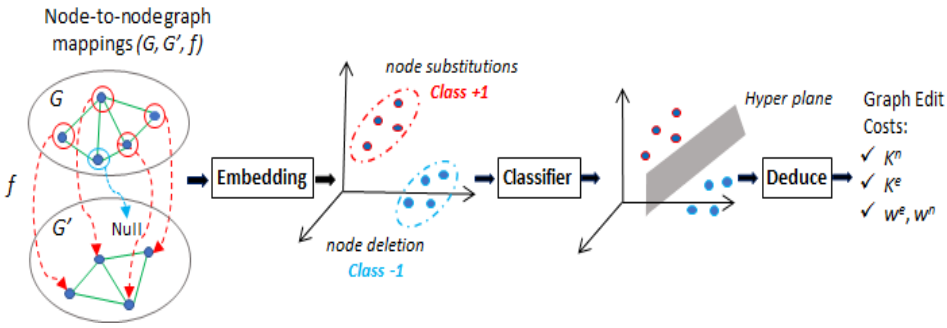


Fig 3. 2: Basic scheme of the of line learning method.

3.3 The Learning Algorithm

Algorithm 1 computes w_v, w_e, K_v and K_e given a learning set that is composed of elements of the form$(G^p, G^q, f^{p,q})$. A graph database with this structure was presented in [56]. In Figure 3.2 represent the basic scheme of learning method.

We need the points in the Euclidean space to be classified. Therefore, we impose the points generated by a substitution to belong to $Class_{a,i}^{p,q} = 1$ and the points generated by a deletion to belong to $Class_{a,i}^{p,q} = -1$. Thus, the hyper plane in equation 1 is deduced by any two-class linear classifier [77] applied to the points in S considering these two classes.

Our learning algorithm is composed of two main parts. In the first part, (from line 1 to line 15) the whole edit operations are transformed into classified points in S. In the second part, (from line 16 to line 21) the hyper plane that defines the border between classes in both systems is learned and the final parameters are deduced. Firstly, the sets are prepared as a union of all the points and secondly, the final data S is generated as a concatenation of these sets. And thirdly, the linear weights are learned though function Classifier.

Line 1 considers all elements in the learning database composed by the tripled($G^p, G^q, f^{p,q}$). Then, all mappings in each tripled are considered (line 2 and 3). In cases where the mapping is a substitution (line 4), a point in the Euclidean space is generated (line 6) and classified (line 7).

In cases where the mapping is a deletion (line 8), several points are generated (line 11) and classified (line 12). These points that represent deletions are generated through the rest of the nodes in G^q that are not null. As commented before, the insertion cases are not considered even though a modification of the algorithm to consider this case would be trivial. Nevertheless, we consider it is not necessary if in the database we always impose $f^{q,p}$ to be the inverse of $f^{p,q}$. This is because while inverting a correspondence the insertions become deletions and vice versa.

Algorithm 1

1. **For all** elements $(G^p, G^q, f^{p,q})$ in the learning set
2. **For** $1 \leq a \leq n^{p,q}$
3. $v_i^q = f^{p,q}(v_a^p)$
4. **If** $v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p$ and $v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q$ (**Substitution**)
5. **If** $\mathrm{T_{a,i}} \neq \mathrm{n}_{\mathrm{N_a^p}} + 1$
6. Compute $\forall t$: $S_{v\,a,i(t)}^{p,q}$, $S_{e\,a,i(t)}^{p,q}$ & $S_{k_e\,a,i}^{p,q}$ Eq.3.6
7. Set $Class_{a,i}^{p,q} = 1$
8. **If** $\mathrm{v_a^p} \in \Sigma_{\mathrm{v}}^{\mathrm{p}} - \hat{\Sigma}_{\mathrm{v}}^{\mathrm{p}}$ and $\mathrm{v_i^q} \in \hat{\Sigma}_{\mathrm{v}}^{\mathrm{q}}$ (**Deletion**)
9. **For all** $\mathrm{v_{i\prime}^q} \in \Sigma_{\mathrm{v}}^{\mathrm{q}} - \hat{\Sigma}_{\mathrm{v}}^{\mathrm{q}}$
10. **If** $\mathrm{T_{a,i\prime}} \neq \mathrm{n}_{\mathrm{N_a^p}} + 1$
11. Compute $\forall t$: $S_{v\,a,i\prime(t)}^{p,q}$, $S_{e\,a,i\prime(t)}^{p,q}$ & $S_{k_e\,a,i\prime}^{p,q}$ Eq.3.8
12. Set $\mathrm{Class_{a,i\prime}^{p,q}} = -1$
13. **End For all**
14. **End For**
15. **End For all**
16. $\forall t$: $S_{v(t)} = \cup\, S_{v\,a,i(t)}^{p,q} \forall a, i, p, q$
17. $\forall t$: $S_{e(t)} = \cup\, S_{e\,a,i(t)}^{p,q} \forall a, i, p, q$
18. $S_{\mathrm{k_e}} = \cup\, S_{\mathrm{k_e}\,a,i}^{p,q} \forall a, i, p, q$
19. $Class = \cup\, Class_{a,i}^{p,q}\ \forall a, i, p, q$
20. $S = \left[S_{v(1)}, \ldots, S_{v(N)}, S_{e(1)}, \ldots, S_{e(M)}, S_{k_e}\right]$
21. $[\mathrm{w_v}, w_{\mathrm{e}}, \mathrm{K_v}, \mathrm{K_e}] = \mathrm{Classifier}(S, Class)$

End algorithm

3.4 Experimental Validation

The method we present has been compared to the only method that learns the insertion and deletion costs [50] to do so, we have used the tarragona-graph repository detailed in [56] (Table 3.1).

The tarragona-graph repository has the main characteristic that each register is not only composed of a graph and its class. Contrarily, each register is composed of a pair of graphs, a ground-truth correspondence between them (mapping between their nodes), as well as their class. This repository was designed to analyze and develop graph-matching algorithms and also learning algorithms designed to learn the graph-matching parameters in a broadly manner. Letter Low, letter Med and letter High are three graph databases that represent artificially distorted letters of the Latin alphabet.

The difference between them is the distortion degree. Rotation-Zoom database is composed of graphs extracted from outdoor images. Nodes are salient points detected by the SIFT method and edges are computed applying Delaunay triangulation on the position of the salient points. Finally, Palmprint database is composed of graphs that have been extracted from palmprints. Nodes are minutiae and edges are computed applying Delaunay triangulation on the position of these minutiae.

In the first three rows of Table 3.1, we show the number of graphs. In the following three rows, we show the number of correspondences between pairs of graphs that belong to the same class. These correspondences

represent the ground-truth node-to-node mapping between graphs. In the case of the rotation-zoom database, the ground truth has been computed through the homography of the images.

In the case of the Palmprint database, it has been computed using the Matlab fingerprint matching function. Finally, in the case of the letter databases, the ground-truth has been defined by construction of the database, since the generation of the letters and distortion of them has been done artificially. Not all combination of correspondences between graphs of the same class have been included in the databases.

Database		Rotat. Zoom	Palm print	Letter		
				Low	Med	High
Graphs	Train	20	80	750	750	750
	Valid.	10	0	750	750	750
	Test	20	80	750	750	750
Corresp.	Train	80	320	37500	37500	37500
	Valid.	40	0	37500	37500	37500
	Test	80	320	37500	37500	37500
Classes		5	20	15	15	15
Attributes		66	5	2	2	2
Description		SIFT	x, y, θ	(x,y)		
Avg. Nodes		50	836.3	4.6	4.6	4.6
Avg. Edges		277.4	4971.2	6.2	6.4	9
Avg. Null mapping		31.6	152.1	0.4	0.4	0.4
Max. Nodes		50	1505	8	9	9
Max. Edges		284	8962	12	14	18
Max. Null mapping		50	619	4	5	5

Table 3. 1: Main database features

Table 3.2 shows the matching accuracy of these methods in the different databases. In the case of [56] the authors claim the results are the best ones obtained by manually tuning the parameters. The matching accuracy is computed as the inverse of the hamming distance between the ground-truth correspondence $\hat{f}^{p,q}$ and the automatically deduced correspondence $\dot{f}_{sub}^{p,q}$,

[56] Table 3.3 and Table 3.4 show the learning runtime of these methods in the different databases. Due to runtime restrictions, palmprint with [50] has not been tested. Our method has been tested considering only the insertion and deletion costs and the whole weights.

This comparison has been proposed to show how important is to properly tune these weights. The Matlab code of these experiments is available in [57] more than the ones manually tuned [56] ratios. The method in [50] turns out to be very slow. Note that the runtime of this method depends on the order of the graphs, which is not the case of our method.

	[56]	[50]	Our method	
			Only Kv, Ke	All the weights
Letter High	%83	%83	%88	%91
Letter Med	%83	%88	%90	%94
Letter Low	%90	%91	%92	%99
Rot. Zoom	%85	%24	%80	%85
Palmprint	%47	---	%48	%53

Table 3. 2: Matching accuracy

	[50]	Our method	
		Only Kv, Ke	All the weights
Letter High	33	0.19	0.26
Letter Med	47	0.18	0.30
Letter Low	42	0.22	0.29
Rotation Zoom	821	8.6	510
Palmprint	---	38	66

Table 3. 3: Learning runtime in seconds

Figures 3.3- 3.6 show the coordinate system S of the four databases that have two attributes (more than two attributes cannot be visualized). Almost all the edit operations are properly classified considering the plane that our method has learnt.

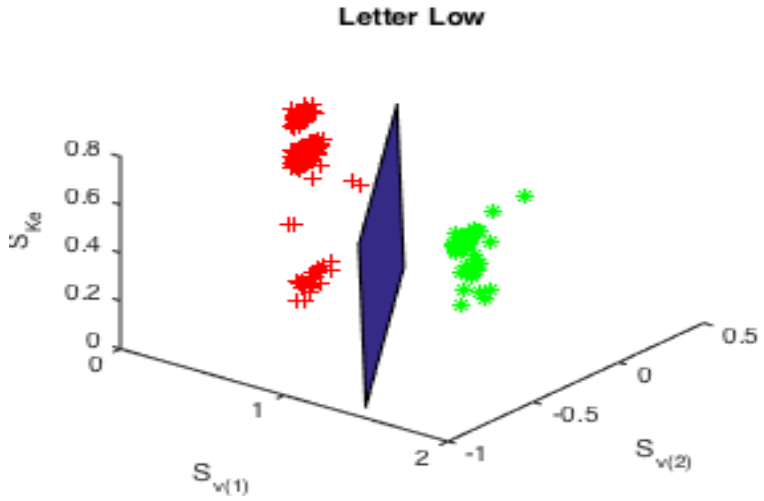


Fig 3. 3: Coordinate system: S. +: Substitutions. *: Deletions. Of used Database :letter low

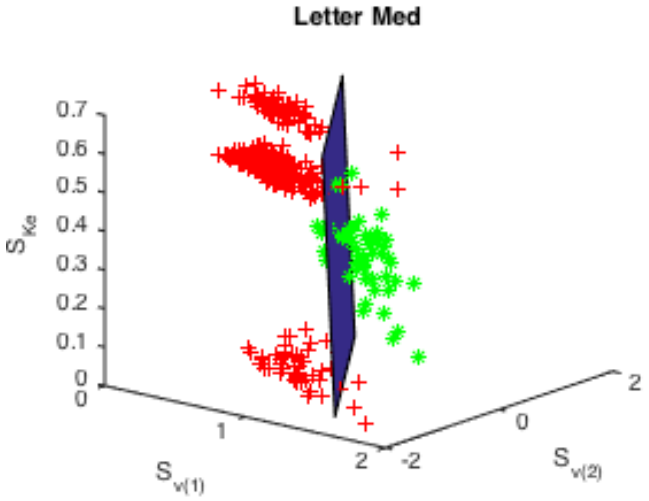


Fig 3. 4: Coordinate system: S. +: Substitutions. *: Deletions. Of used Database: letter Med

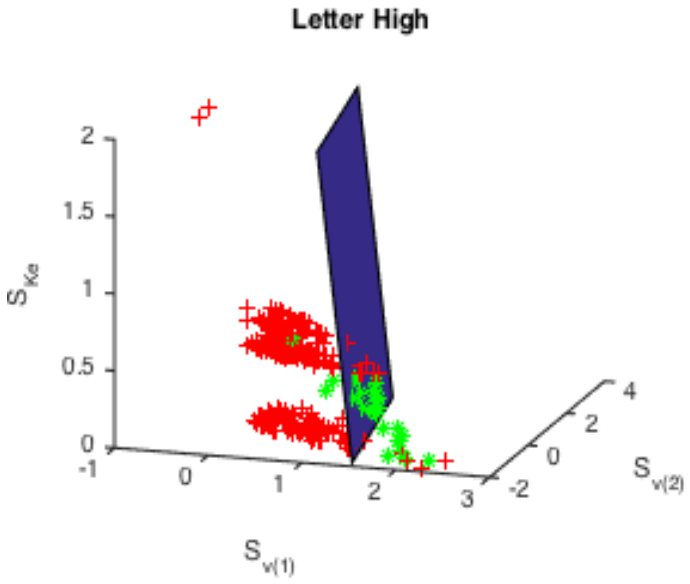


Fig 3. 5: Coordinate system: S. +: Substitutions. *: Deletions. Of used Database : letter High

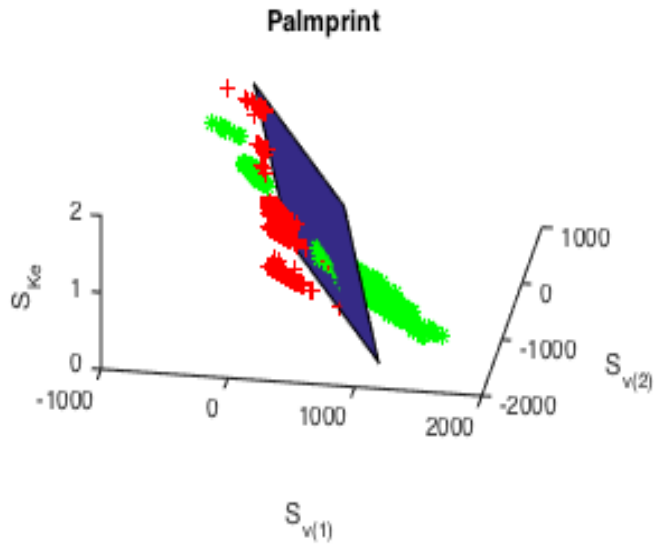


Fig 3. 6: Coordinate system: S. +: Substitutions. *: Deletions . Of used Database : Palmprint

	[50]		Our method				
			Only Kv,Ke		All the weights		
	K_v	K_e	K_v	K_e	K_v	K_e	w_v
L. High	1.65	-0.31	1.94	-0.02	-1.26	0.06	0.75 0.25
L. Med	2.32	-0.16	1.54	-0.07	-1.03	0.37	0.86 0.14
L. Low	1.29	-0.21	0.65	-0.12	-0.70	0.11	0.16 0.84
Rot. Zoom	0.006	-0.03	4.18	-4.71	-0.006	0.008	**Fig.3.7**
Palmprint	---	---	670	-277	-328	136	0.49 0.51

Table 3. 4: Learned GED parameters.

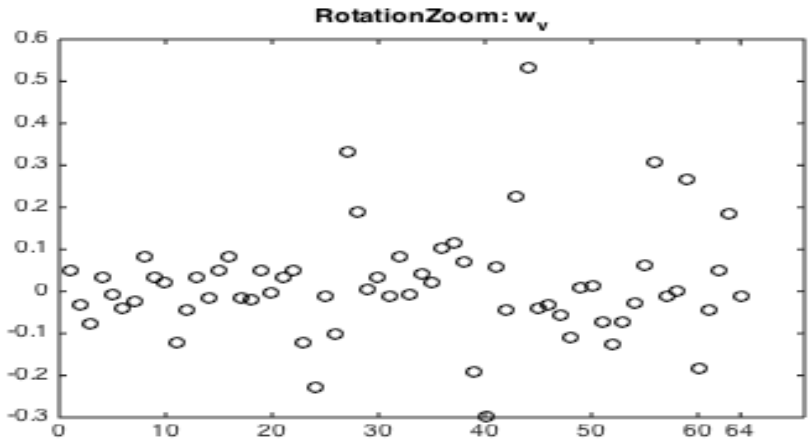


Fig 3. 7: shows the learnt parameter w_v of the Rotation Zoom.

Fig. 3.7 Vector of weights w_v of the Rotation Zoom database. Attributes on nodes of this database are composed of a 64-cell vector of SIFT descriptors. The addition of the 64 values equals 1, which is imposed by our method.

3.5 Application to Point Image Registration

3.5.1 Introduction

Alignment of point sets is frequently used in pattern recognition when objects are represented by sets of coordinate points. The idea behind this problem is to be able to compare two objects regardless of the effect of a given transformation on their coordinate data. This method presents a align point sets based on the graph edit distance. The main idea is to learn the edit costs (in a learning step) and then apply graph edit distance (in a pattern recognition step) with the learned edit costs. Thus, the edit cost would have to incorporate the transformation parameters. In the experimental section, we show that the method is competitive if the graph edit distance parameters are automatically learned considering the learning set. These parameters are the insertion and deletion costs and also the weights on the substitution costs.

The core of many object recognition applications, in which objects are defined by coordinate data, is the point set alignment. For instance, medical image analysis or reconstructing a scene from various views [58].

The idea behind it is to be able to compare two objects regardless of the effects of a given transformation on their coordinate data.

The aim of this method is to present a method for point set alignment based on the graph edit distance. Recently, Deep learning techniques have been used to perform image registration. Nevertheless, it has to be considered that these techniques can only be properly used in huge databases, which it is not always the case. For this reason, we consider it is worth analyzing old methods and presenting new ones based on point-set alignment. Given that the point correspondences are known, some years ago, there was an extensive work done towards the goal of finding the alignment parameters that minimize some error measure.To cite a few, [59,60] deal with isometries and similarity transformations; [61,62] deal with Euclidean transformations (i.e. excluding reections from isometries); [63] deals with similarity and projective transformations; and [58] deals exclusively with projective transformations. However, the point-set alignment problem is often found in the more realistic Setting of unknown point-to-point correspondences. This problem becomes then a registration problem, that is, one of jointly estimating the alignment and correspondence parameters. Although non-iterative algorithms exist for specific types of transformation models [64], this problem is usually solved by means of non-linear iterative methods that, at each iteration, estimate correspondence and alignment parameters. Despite being more computationally demanding, iterative methods are more appealing to us than the direct ones due to its superior tolerance to noise and outliers.

We distinguish between two families of approaches to solve this problem. Ones are based on the Expectation Maximization (EM) algorithm [65], and the others use Softassign [66,67,68] . The former ones have the advantage of offering statistical insights of such decoupled estimation processes while the latter ones benefit from the well-known robustness and convergence properties of the Softassign embedded within deterministic annealing procedures.

Coherent Point Drift (CPD) [69] is a point-set registration method that uses the EM algorithm that is defined for rigid, affine and non-rigid transformations. Moreover, Robust Point Matching (RPM) [67,68] is a method that uses Softassign that is defined for affine and rigid transformations. Later, TPS-RPM [70] was presented, which is an extension to non-rigid transformations of RPM.

Graph matching approaches allow for neighboring relations between points to be considered into the point-set registration problem. Graduated assignment [66] and bipartite graph matching [62,30,71,24] are remarkable graph matching methods that use soft assign and least sum assignment solvers. An approach for graph matching and point-set alignment using the EM algorithm that was defined for affinities and projectivities was presented in [72] .One limitation of this approach is the high computational demand of the dictionary-based structural model. In [73], it was proposed an EM-like approach for graph matching and point-set alignment based on a cross-entropy measure. They proposed a model of structural errors based on a Bernoulli distribution. This model was defined as rigid-body transformations.

Finally, [74] proposed a joint structural graph matching and point-set registration method whose main contribution was to bridge the gap between the EM-based and the Softassign-based approaches by formulating the graph matching problem within a principled statistical framework, while beneting from the desirable properties of the Softassign and deterministic annealing ensemble.

The aim of this method is to present an image registration method based on the graph edit distance [8,38], That is, we present a classical graph matching approach based on a well-known distance between graphs. The novelty of the method is not the transformation of the image into an attributed graphs neither the graph edit distance itself. Contrarily, the novelty is to show a method that has the capability of learning the parameters of the graph edit distance such that the image registration is properly carried out. In the past, any thesis was presented that learned the graph edit distance parameters and directly applied the graph edit distance to deduce the salient points correspondences.

We assume it was due to the graph edit distance parameters were manually tuned and then, the deduced graph correspondence was far away from the optimal one. This was the reason why other mechanisms were presented that, in the process itself, somehow related the graphs into the images, for instance methods [73,72,74].

Note that the information on the salient points (and thus, on the nodes of the graphs) is usually composed of a vector larger or equal to 60 elements. This feature makes impossible to manually tune the weights on each vector element.

	Boat	East park	East sout	Residence
Sanroma	48	49	25	25
Unfied	150	130	120	135
Dual step	54	35	28	28
Graduated assignment	175	120	126	120
Matching by correlation	185	130	130	150
Neural network	36	30	19	27
Graph edit distance	0	0	0	0

Table 3. 5: Mean Projection Error on each database

3.5.2 Experimental Validation of the application

Four databases have been used to test the proposed method and compare it to known image registration methods: Boat, East park, South park and Residence. These databases are composed of sequences of images, in which there is the same object but the point of view has been slightly modified. From these databases, we have extracted 50 salient points per each image using the SURF extractor. Moreover, we have generated attributed graphs that nodes are the salient points and edges have been deduced by Delaunay triangulation. The attributes on the nodes are the

information deduced by the SURF extractor (a vector of 60 elements) and edges are unattributed.

Finally, a ground-truth correspondence between the salient points of adjacent images in the sequences have been manually deduced, for testing purposes. Not all the 50 salient points are present in all the images in the sequence, for this reason, the ground-truth correspondences have Substitutions (a salient point of the first image is mapped to a salient point of the second image), deletions (the salient point of the second image is not present) and insertions operations (the salient point of the first image is not present). The databases composed of these sequence of graphs are available at [57] and an extended explanation of the databases were published in [56]. Figure 3.8 shows an image of each database with their attributed graph.

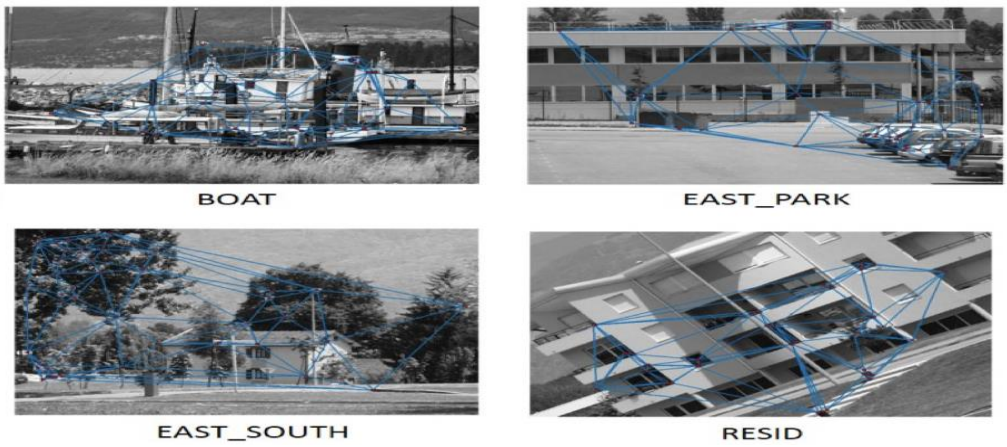


Fig 3. 8: An example of an image per database

Using these databases, the learning algorithm in [44] was able to learn the substitution costs C_s , the insertion costs C_i and the deletion costs C_d . Then, we analyzed the ability of the graph edit distance to deduce the salient point correspondence between pair of images, given these costs. Recall that the point correspondence is the first step to perform image registration. The point correspondence deduced by the graph edit distance was compared to classical methods, such as Sanroma [74], Unifed [73], Dual step [72] Graduated assignment [67] Matching by correlation [72], Neural Network [53] and Algabli [44]. The quality of the point correspondences has been set through two metrics:

1. The mean projector error which is the mean projector error is a quality measure that informs of the mean Euclidean distance between the receiving pixel of the ground truth and the receiving pixel of the correspondence deduced by the matching algorithm, given all the mapped points of all pairs of images.

2. The number of mapped points is a simpler measure, which informs of how many points on the first image are mapped to points on the second image. Note that in this case, the ground-truth correspondence does not influence on this measure.

Table 3.5 shows the mean projector error deduced by each algorithm using the previously commented databases. We realize there are extreme values. For instance, the graph edit distance with the edit costs

learned by Algabli [44] (the method we wanted to analyse) generates zero error but Matching by correlation [72] generates an error between 130 and 185. Having seen these results, it is important also to have a look at Table 3.6, which shows the mean number of mapped points per pair of images. Note that the last row refers to this measure given the ground truth correspondence. Thus, if an algorithm returns a mean number of mapped points lower than the ground truth number, it means that the algorithm tends to map too few points or discard good ones. Conversely, if an algorithm returns a mean number higher than the ground truth number, it means that the algorithm tends to map too much points, and thus, it maps points that would not have to be mapped. Comparing the selected algorithms, on the one side, we realize that the graph edit distance generates a zero mean projection error but it also generates few number of mapped points. Thus, this algorithm only returned the mapped points with a high confidence. On the other side, Unified [73], Graduated assignment [66] and Matching by correlation [72] mapped all the points and, for this reason, they generated the highest mean projection error.

We conclude the quality of these matching algorithms cannot be analyzed without considering the mean projection error and also the mean number of mapped points. Moreover, it is usually preferable to return a low number of mapped points but having a good quality (low mean projection error) than to return a larger number of mapped points but increasing the number of wrong ones. This is because, only some mapped points is enough to deduce the image transformation. Besides, the

algorithms that approximate the image transformations usually return better results with few non-noisy points than with a larger number of points, which some of them being noisy.

	Boat	East park	East sout	Residence
Sanroma	29	38	37	35
Unfied	50	50	50	50
Dual step	35	32	33	35
Graduated assignment	50	50	50	50
Matching by correlation	50	50	50	50
Neural network	18	23	17	22
Graph edit distance	9.24	9.6	9.48	10
Ground truth	17.44	16.16	12.80	17.56

Table 3. 6: Mean number of mapped points on each database.

Chapter 4: Learning the Fingerprint Distance

4.1 Introduction

In this chapter, we show a new method to learn the weights w_{xy} and w_{α} based on the experience of the fingerprint specialists. Humans are good at selecting the same physical point in two different images at a somewhat high level. In the case of fingerprint identification, specialists map minutiae from a pair of fingerprints very accurately and fast. For this reason, we have implemented a friendly interface to map minutiae of pairs of fingerprint, which are going to be considered to be the best correspondence for our learning purposes. Figure 4.1 shows the output of our Interface. Through this interface, the human maps only some pairs of minutiae by simply clicking the mouse on the position of the minutiae. As it is shown in the experimental section, few minutia-to-minutia mappings are needed to properly learn these weights. Moreover, we suppose the interface does not allow to map two minutiae that have different endings, terminal or bifurcation, to avoid having the correspondence an infinity cost Equation 2.12 in chapter 2. We call $\hat{f}^{p,q}$ as the human-imposed correspondence between fingerprints F^p and F^q.

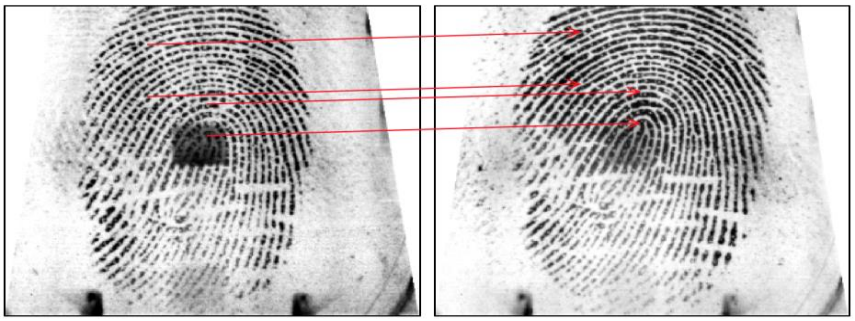


Fig 4. 1: Output of the fingerprint correspondence editor. The specialist has imposed four minutia-to-minutia mappings.

We want to learn the weights w_{xy} and w_{α} through a supervised learning method. The aim of the method is to obtain these weights so that the Automatically-deduced correspondences $\dot{f}^{p,a}$ become close to the human-made correspondences $\hat{f}^{p,q}$ for all pairs of fingerprints(F^p, F^q).

In this way, the human-made correspondences become the ground-truth correspondences in the learning process. It is important to note that these Ground-truth correspondences are independent of these weights w_{xy} and w_{α} since it is an expert who decides them only by looking at the fingerprint image.

Given a specific mapping between a pair of minutiae within two fingerprints imposed by the specialist, we could assume that these two minutiae are more similar than the rest of combinations between the minutia of the first graph and the other minutiae of the second fingerprint. This is the cornerstone of our learning method: The specialist, without

considering weights neither distances, decides the best correspondence and discards the other combinations. Nevertheless, and due to the noise generated by each fingerprint scanning, we realised that imposing the selected minutia-to-minutia mapping to have the minimum distance is too restrictive. Then we decided to assume that this selected mapping is similar to the average of the other combinations.

Let us be more specific. Assume we have two fingerprints F^p and F^q and that the specialist has decided that $\hat{f}^{p,q}(m^a) = m^i$, where $m^a \in F^p$ and $m^i \in F^q$. Then, our learning method is based on assuming that,

$$MD_{w_{xy},w_\alpha}(m^a, m^i) = \min_{\substack{\forall m^j \in F^q \; s.t. \\ d_{TB}(m^a_{TB}, m^j_{TB})=0}} \left\{ MD_{w_{xy},w_\alpha}(m^a, m^j) \right\} \qquad (4.1)$$

Which means that the mapping selected by the specialist is the one that has the average distance considering all the combinations from m^a to any minutia in F^a.Figure 4.2 shows a selected minutia-to-minutia mapping in a continuous line (the left part of Equation 4.1) and the rest of the mappings in dashed lines (the right part of Equation 4.1). Note that the left part is also included in the right part of the Equation.

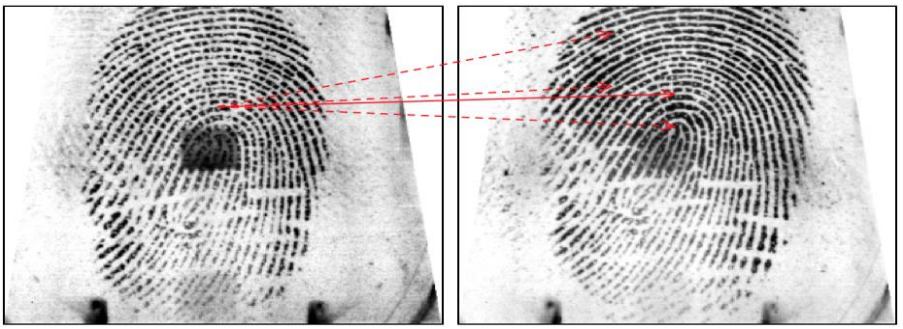


Fig 4. 2: Continuous arrow: Imposed mapping. Dashed arrows: the other mapping combinations from the mapped minutia of the first fingerprint.

4.2 Embedding the minutia-to-minutia mappings

Our learning method is based on two steps. First, embedding all the ground truth minutia-to-minutia mappings into a Euclidean 2D space$S = (S_{xy}, S_{\alpha})$. Second, finding the linear regression (a line in a 2D space) of the set of points to deduce the weights w_{xy} and w_{α}. This is because the Space S is defined such that the constants of the plane that forms the regression are exactly the weights w_{xy} and w_{α}.We move on the definition of S. If we want Equation 4.1 to hold for all minutia-to-minutia mappings imposed by the human, then we have that

$$\forall m^{a\,p} \in F^p:$$

$$\min_{\substack{\forall m^j \in F^a \, s.t. \\ d_{TB}\left(m^a_{TB}, m^j_{TB}\right)=0}} \begin{matrix} w_{xy} \cdot d_{xy}\left(m^a_{xy}, m^i_{xy}\right) + w_\alpha \cdot d_\alpha\left(m^a_\alpha, m^i_\alpha\right) - \\ \left\{w_{xy} \cdot d_{xy}\left(m^a_{xy}, m^j_{xy}\right) + w_\alpha \cdot d_\alpha\left(m^a_\alpha, m^j_\alpha\right)\right\} = 0 \end{matrix} \qquad (4.2)$$

Where m^i represents the specialist imposition for each m^a. That is, $\hat{f}^{p,a}(m^a) = m^i$. As previously commented, the mappings that generate infinity distance in d_{TB} are discarded. Then, with some arrangements, we arrive at the following expression,

$$\forall m^a \in F^p:$$

$$w_{xy} \cdot \left(d_{xy}\left(m^a_{xy}, m^i_{xy}\right) - \min_{\substack{\forall m^j \in F^q \, s.t. \\ d_{TB}\left(m^a_{TB}, m^j_{TB}\right)=0}} \left\{d_{xy}\left(m^a_{xy}, m^j_{xy}\right)\right\} \right) +$$

$$w_\alpha \cdot \left(d_\alpha\left(m^a_\alpha, m^i_\alpha\right) - \min_{\substack{\forall m^j \in F^q \, s.t. \\ d_{TB}\left(m^a_{TB}, m^j_{TB}\right)=0}} \left\{d_\alpha\left(m^a_\alpha, m^j_\alpha\right)\right\} \right) = 0 \qquad (4.3)$$

Considering this expression, we embed a concrete mapping $\hat{f}^{p,a}(m^a) = m^i$, into the space $S = (S_{xy}, S_\alpha)$ as a point $\left(S^a_{xy}, S^a_\alpha\right)$ as follows,

$$S_{xy}^{a} = d_{xy}\left(m_{xy}^{a}, m_{xy}^{i}\right) - \min_{\substack{\forall m^{j} \in F^{q}\ s.t. \\ d_{TB}\left(m_{TB}^{a}, m_{TB}^{j}\right)=0}} \left\{d_{xy}\left(m_{xy}^{a}, m_{xy}^{j}\right)\right\}$$

$$S_{\alpha}^{a} = d_{\alpha}\left(m_{\alpha}^{a}, m_{\alpha}^{i}\right) - \min_{\substack{\forall m^{j} \in F^{q}\ s.t. \\ d_{TB}\left(m_{TB}^{a}, m_{TB}^{j}\right)=0}} \left\{d_{\alpha}\left(m_{\alpha}^{a}, m_{\alpha}^{j}\right)\right\} \quad (4.4)$$

Finally, if we consider Equation 4.3 and Equation 4.4, it is simple to realise that the whole points have to hold the following property,

$$\forall m^{a} \in F^{p}:$$

$$w_{xy} \cdot S_{xy}^{a} + w_{\alpha} \cdot S_{\alpha}^{a} = 0 \quad (4.5)$$

Nevertheless, do to data inconsistency and noise, it is unlike that Equation 4.5 holds for all the imposed mappings. In this way, the set of Minutia-to-minutia mappings becomes a cloud of points in S instead of the line represented in Equation 4.6.

$$w_{xy} \cdot S_{xy} + w_{\alpha} \cdot S_{\alpha} = 0 \quad (4.6)$$

Thus, our aim is to get as much close as possible to this line, which best fits the cloud of points. Thus, the computed weights would have to minimise the following functional,

$$\sum_{\substack{\forall m^{j} \in F^{q}\ s.t. \\ d_{TB}\left(m_{TB}^{a}, m_{TB}^{j}\right)=0}} \left(w_{xy} \cdot S_{xy}^{a} + w_{\alpha} \cdot S_{\alpha}^{a}\right)^{2} \quad (4.7)$$

4.3 The Learning Algorithm

We learn the weights w_{xy} and w_{α} by finding the regression line in Equation 4.7 that has the two following properties: 1) it has a point in the origin $(0,0)$ since there is no offset in Equation 2.11 in chapter 2. It holds the restriction in Equation 2.14 in chapter 2.These two properties allow us to apply the Lagrange multipliers [76] to deduce the weights. Since now, we have assumed that, for the learning purposes, we have only a pair of fingerprints F^p and F^a and its ground-truth correspondence$\hat{f}^{p,a}$. Nevertheless, our method allows us to have several pairs of fingerprints with their correspondences. Thus, the learning set is composed of T triplets of the form$\left(F^p, F^q, \hat{f}^{p,q}\right)$. A graph database with this structure was presented in our learning algorithm is composed of the following five steps [56].

First, define the T following matrices SM^p that have two columns and some rows for all triplets$\left(F^p, F^q, \hat{f}^{p,q}\right)$. In the experimental section, we have seen that the number of rows, that is, the number of minutia-to-minutia impositions could be much smaller than$|F^p|$.

$$SM^p = \begin{bmatrix} S_{xy}^{1\ p} & S_{\alpha}^{1p} \\ \dots & \dots \\ S_{xy}^{|F^p|p} & S_{\alpha}^{|F^p|p} \end{bmatrix} \tag{4.8}$$

Each expression$S_{xy}^{1\ p}$, …, $S_{xy}^{|F^p|^p}$ and $S_{\alpha}^{1\ p}$, …, $S_{\alpha}^{|F^p|^p}$ is computed through Equation 4.4 given the triplet $\left(F^p, F^q, \hat{f}^{p,q}\right)$.

Second, concatenate the T matrices in Equation 4.8 as follows,

$$SM = \begin{bmatrix} SM^1 \\ \dots \\ SM^T \end{bmatrix} \quad (4.9)$$

Third, compute the following symmetric 2x2 matrix (´ represents the transposed):

$$\Sigma = SM' \cdot \mathrm{SM} \quad (4.10)$$

Four, deduce the eigenvectors and eigenvalues from matrixΣ.

Five, set the weights w_{xy} and w_{α} as the first and second element of the eigenvector that has the minimum eigenvalue (Lagrange multipliers).In the following paragraph, we deduce the computational cost of our learning algorithm. The computational cost of computing expressions $S_{xy}^{a\ p}$ or $S_{\alpha}^{a\,p}$ is linear with respect to the number of minutiae-to-minutia mappings.

In the worst case, we have$O(|\mathrm{F}^{\mathrm{p}}|)$. Then the computational cost of the first two steps is quadratic with respect to the minutia-to-minutia mappings of a specific triplet and multiplied by the number of triplets $O(T \cdot |\mathrm{F}^{\mathrm{p}}|^2)$. The computational cost of the third step can be approximated by $O(|\mathrm{F}^{\mathrm{p}}|^2)$ since matrices have only two columns. The Fourth and fifth step has a constant and negligible cost since Σ is a 2x2 matrix. All in all, we consider the computational cost of our algorithm to be $O(T \cdot |\mathrm{F}^{\mathrm{p}}|^2)$ in the worth case, which is the cost of the first step.

4.4 Experimental Validation

To test our learning method, we have used the $Tarragona\_Fingerprint$ repository available at [57] this repository is composed of 15 different databases, which have the learning, validation and test set. Moreover, the main characteristic of this database is that elements are not classified fingerprints, but triplets composed of two fingerprints from the same individual and a ground-truth correspondence between some minutiae of both fingerprints. There are triplets from different individuals. The format of the database is the one published in [56]. Note that in the $Tarragona\_Fingerprint$ repository, a fingerprint is not represented as

an image but as a set of minutiae. Minutiae attributes are the position, angle and type of minutiae (Terminal or Bifurcation). The fingerprints in $Tarragona\_Fingerprint$ repository were taken from the database [56] and the sets of minutiae were generated given some Matlab libraries. The ground truth correspondence between some minutiae were computed by our correspondence editor. The first three databases in the Repository have been generated as commented. Nevertheless, in the other 12 ones, some controlled noise was added in the position and angle (see Table 4.1). All the functions are available at [57].

The function $\text{gausian\_nose}(\text{Mean}, \text{Max})$ generates a value in a Gaussian probability distribution centred at Mean and having a maximum value of Max. If the value in each cell is X, then the noise added to the position or the angle is $\text{X} \cdot \text{gausian\_noise}(0, \text{Max})$.

Noise	**Pos.**	**Angle**
DB04	0.1	0.1
DB05	0.2	0.2
DB06	0.3	0.3
DB07	0.1	0.4
DB08	0.2	0.1
DB09	0.3	0.2
DB10	0.1	0.3
DB11	0.2	0.4
DB12	0.3	0.1

Table 4. 1: Noise on the position and the angle imposed in the last 12 databases.

Figure 4.3-4.5 shows all the points $\left(S_{xy}^{a}, S_{\alpha}^{a}\right)$ computed through Equation 4 in the embedding space given the first three databases of the $Tarragona\_Fingerprint$ repository (detailed in the practical application Section) with the deduced linear regression (Equation 4.6) described as $S_{xy} = \frac{-w_{\alpha} \cdot S_{\alpha}}{w_{xy}}$.Note axes of these plots are in different scales. For this reason, it seems as the points are clouds but if we drew the plots such that the axes had the same scale, then we'll realize that the points are located in an almost liner cloud .

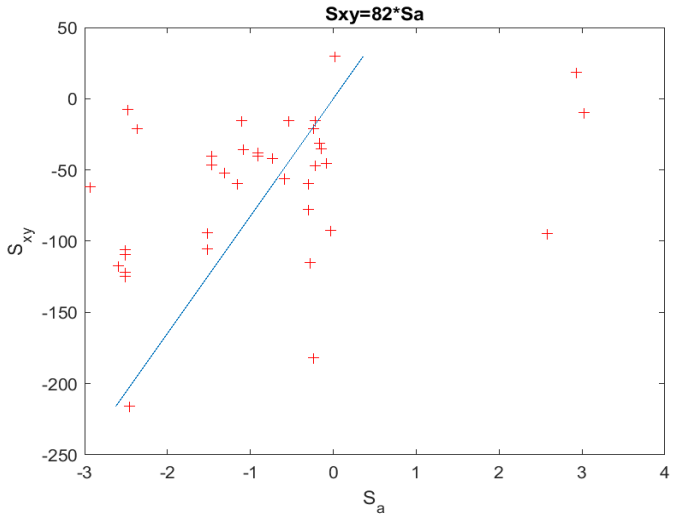


Fig 4. 3: Embedded space and the deduced linear regression of the first databases of the Tarragona_Fingerprint repository.

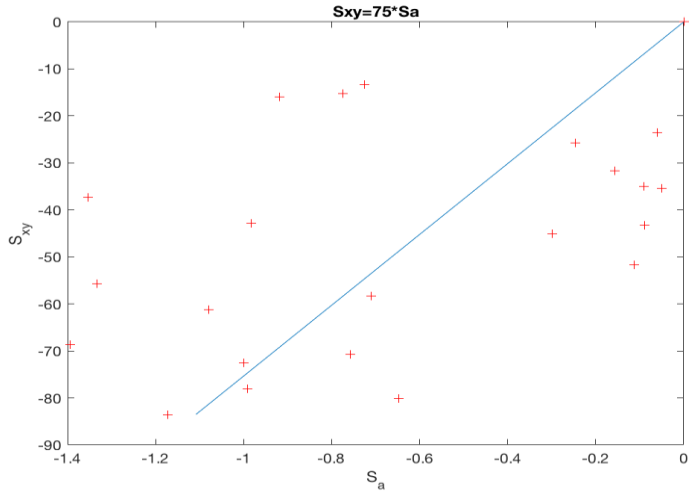


Fig 4. 4: Embedded space and the deduced linear regression of second databases of the Tarragona_Fingerprint repository.

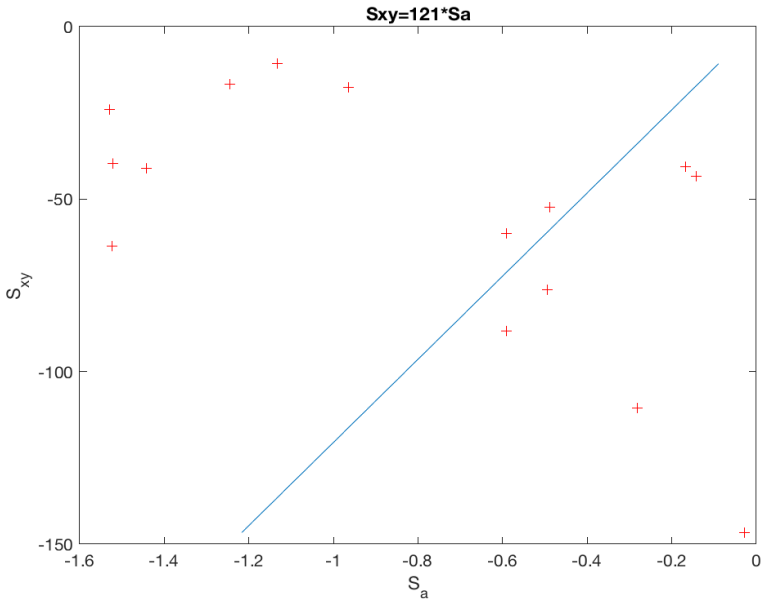


Fig 4. 5: Embedded space and the deduced linear regression of third databases of the Tarragona_Fingerprint repository.

We present two different experiments:

The aim of the first experiments is to analyse the relation between the learned weights and the noise imposed on the position and the angle. Figures 4.3-4.5 show the embedding space deduced from the last databases, in which some noise was added. We realise that when the noise on the angle increases (from top to bottom) the dots in the embedding space tend to move to the right (more positive values in S_α). This is because the distance of the imposed mappings between the angles, $d_\alpha\left(m_\alpha^a, m_\alpha^i\right)$, tends

to increase. Similarly happens when the noise on the position increases (from left to right). In this case, the Euclidean distance of the imposed mappings, $d_{xy}\left(m_{xy}^{a}, m_{xy}^{i}\right)$, tends to increase. Note that the slope of the regression line is dependent on this dots' movement.

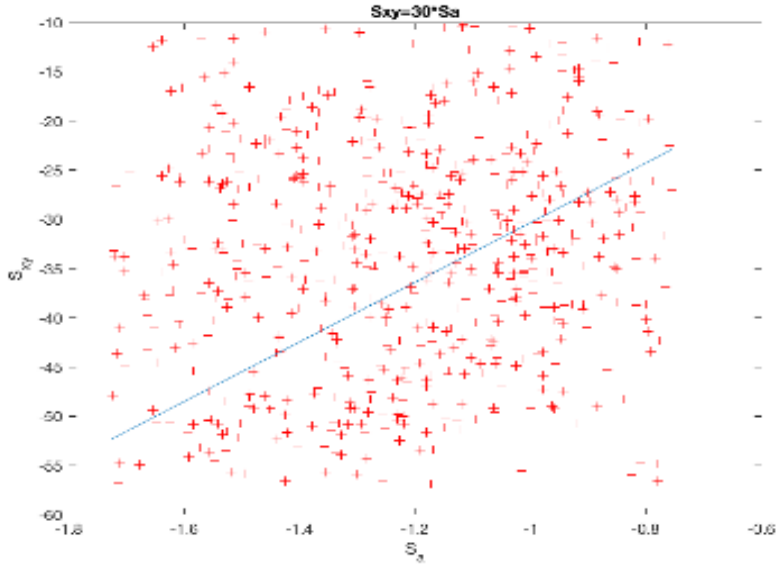


Fig 4. 6: DB04 (NoiseXY: 0.1, NoiseAngle: 0.1)

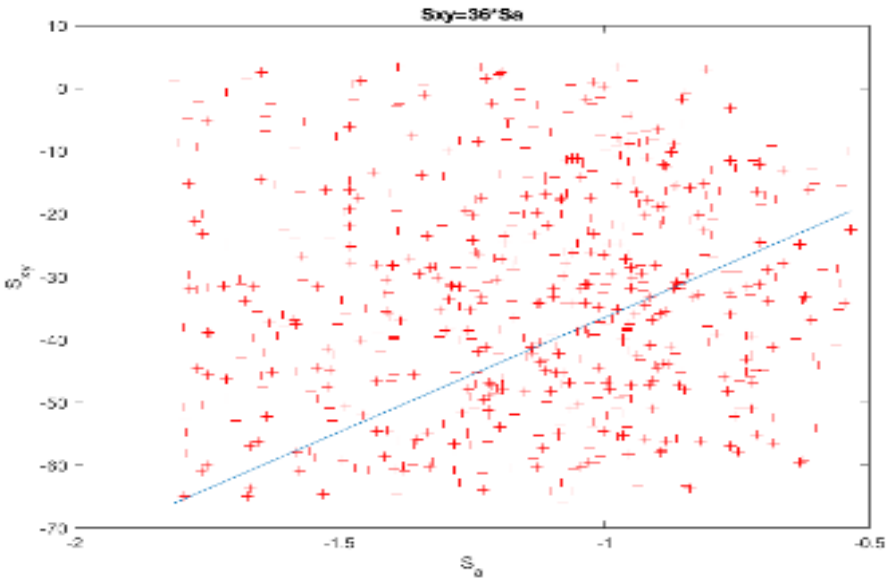


Fig 4. 7: DB08 (NoiseXY: 0.2, NoiseAngle: 0.1)

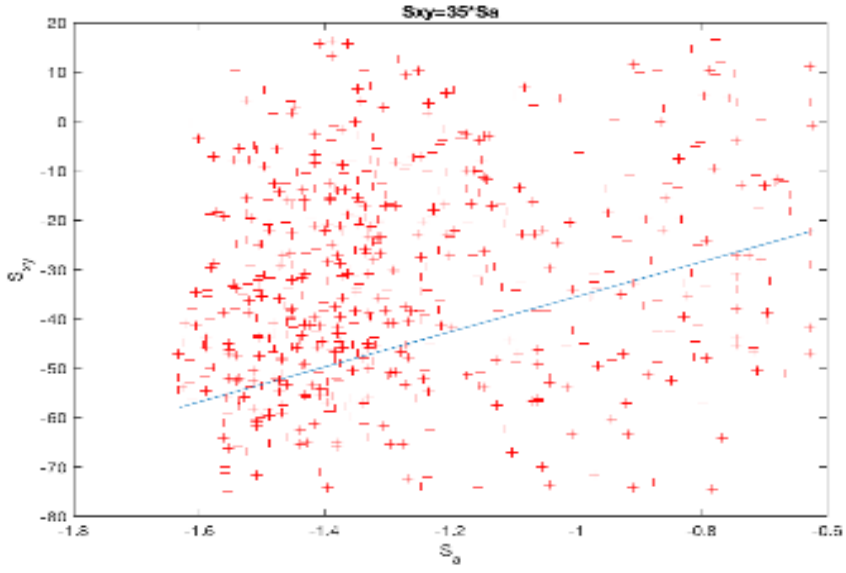


Fig 4. 8:DB12 (NoiseXY: 0.3, NoiseAngle: 0.1)

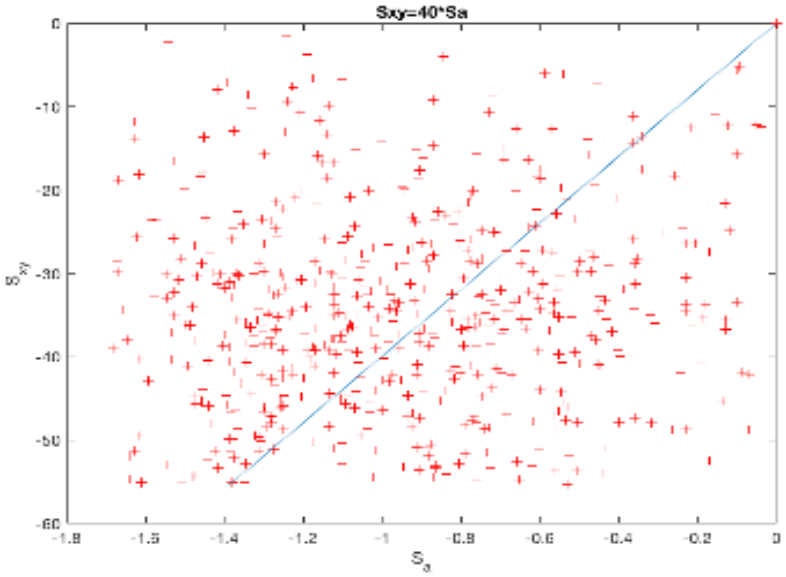


Fig 4. 9: DB05 (NoiseXY: 0.1, NoiseAngle: 0.2)

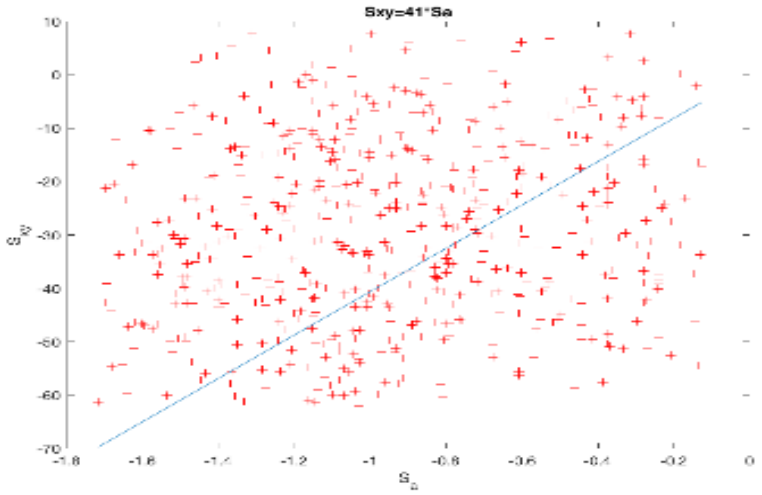


Fig 4. 10: DB09 (NoiseXY: 0.2, NoiseAngle: 0.2)

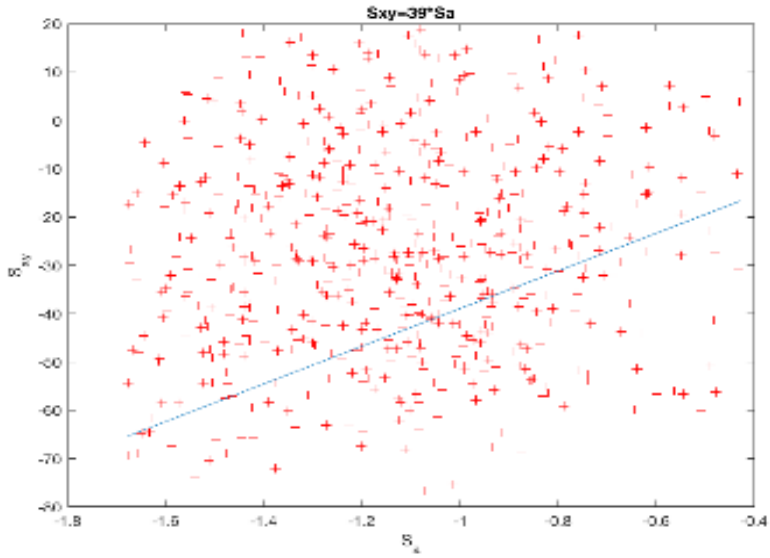


Fig 4. 11:DB13 (NoiseXY: 0.3, NoiseAngle: 0.2)

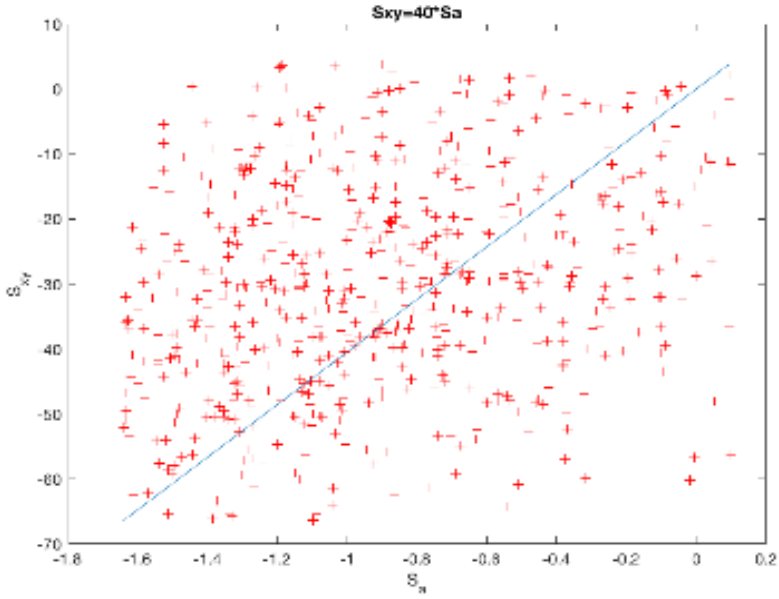


Fig 4. 12:DB06 (NoiseXY: 0.1, NoiseAngle: 0.3)

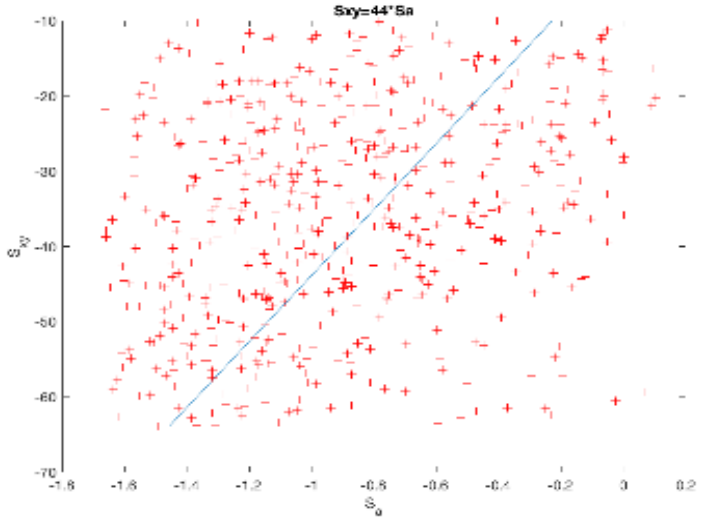


Fig 4. 13: DB10 (NoiseXY: 0.2, NoiseAngle: 0.3)

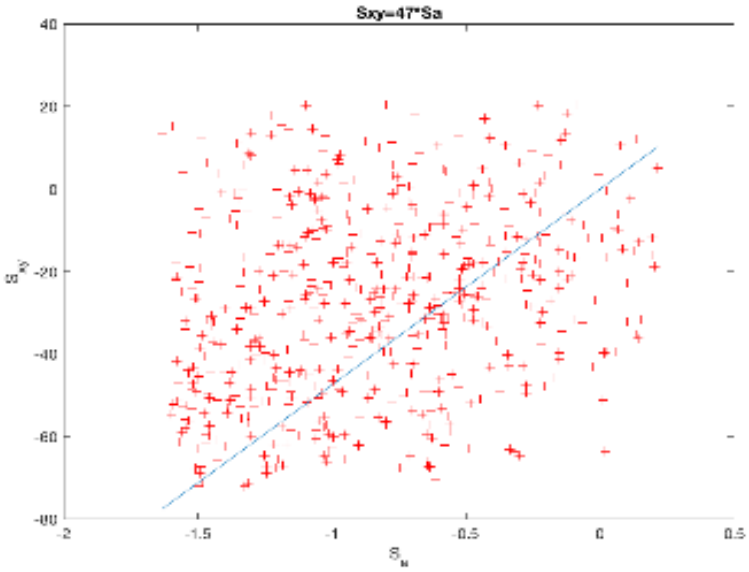


Fig 4. 14: DB14 (NoiseXY: 0.3, NoiseAngle: 0.3)

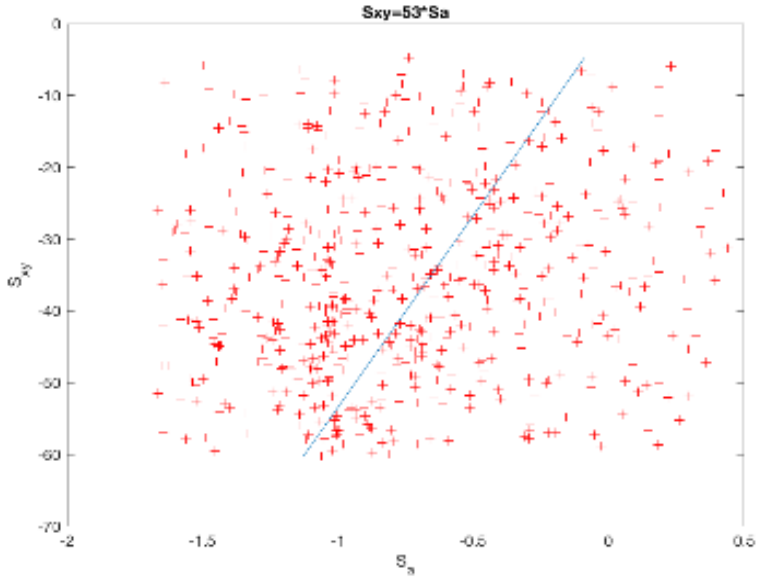


Fig 4. 15: DB07 (NoiseXY: 0.1, NoiseAngle: 0.4)

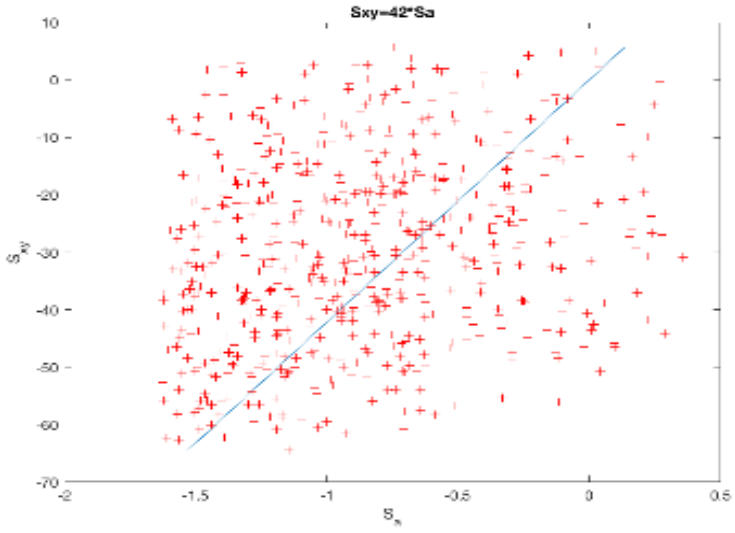


Fig 4. 16: DB11 (NoiseXY: 0.2, NoiseAngle: 0.4)

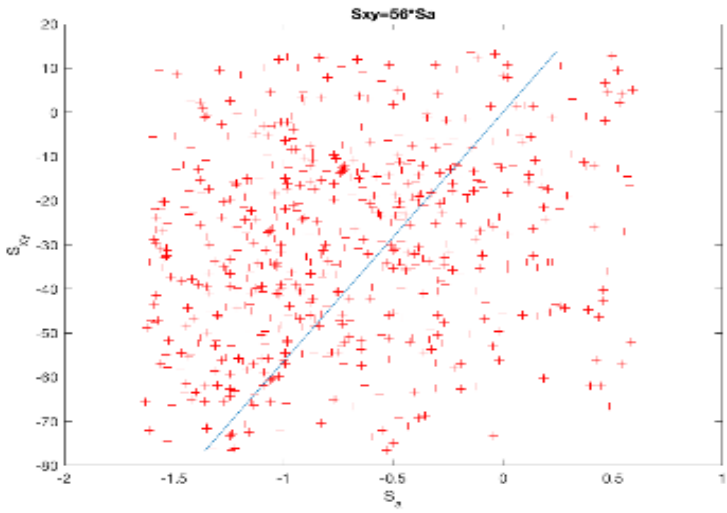


Fig 4. 17:DB15 (NoiseXY: 0.3, NoiseAngle: 0.4)

Fig(4.6- Fig 4.17) Embedded space and the deduced linear regression given the Tarragona_Fingerprint repository.

The aim of the second experiments is to show the classification ratio and the accuracy of our method compared to two other methods. Both methods have been adapted to the fingerprint matching problem. The first one is [46], which is based on minimising the Dunn's index. The second one is inspired in [48] or [80], which are based on minimising the Hamming distance between the ground-truth correspondence and the deduced correspondence. Both methods need an initial value of w_α, and we have tested the value deduced by our method and four other values close to it.

In this way, we supposed the minimisation method could increase our results. Table 4.2 shows the deduced classification ratio (CR) and the Hamming distance (H) between the ground-truth correspondence and the deduced correspondence for the three methods (in an ideal case, the CR would have to be 1 and the HH would have to be 0).

We also show the learned w_α (note that w_{xy} is always set to 1). In the other two methods, we show the results such that the initial values are the ones computed by our method since they were, in general, the best ones. Classification Ratio (CR) and Hamming distance (H) between the ground-truth correspondence and the deduced correspondence given our method (first columns) and two other ones (the rest of the columns) computed in 12 databases. It is also shown the deduced weightw_α. The runtime is shown in the last row

	Linear Regression			[46]			[48]and[80]		
	w_α	CR	H	w_α	CR	H	w_α	CR	H
DB 04	30.31	1	0.01	38.36	1	0.01	30.31	1	0.01
DB 05	39.87	1	0.05	24.56	1	0.04	39.87	1	0.05
DB 06	43.84	0.98	0.06	10.54	1	0.03	26982	0.12	0.66
DB 07	53.45	0.91	0.19	4.19	1	0.08	43839	0.03	0.67
DB 08	36.42	1	0.07	250.81	1	0.08	36.42	1	0.07
DB 09	40.60	1	0.10	142.82	0.92	0.14	41.61	1	0.10
DB 10	40.45	0.9	0.15	37.87	0.91	0.15	49750	0.01	0.63
DB 11	42.14	0.8	0.2	42.14	0.87	0.18	42.14	0.8	0.20
DB 12	35.48	1	0.07	406.14	1	0.06	---	---	---
DB 13	38.95	0.91	0.12	72.33	0.93	0.12	---	---	---
DB 14	47.46	0.78	0.19	55.24	0.77	0.19	48.64	0.78	0.19
DB 15	56.42	0.56	0.25	59.93	0.57	0.25	64.88	0.55	0.26
Average runtime		0.052 sec.			10min.			5min.	

Table 4. 2: Classification Ratio (CR) and Hamming distance (H) computed in 12 databases.

The first we realise is that our method and the one in [46] obtains almost similar results in all the databases when the initial w_α was the one in the second column of the table. In the case of the method similar to [48] or [80], the results are very poor when the noise on the database is high. In two of the cases, the algorithm did not converge, although the initial value was the one in the second column of the table.

The last row of Table 4.2 shows the learning average runtime for the three methods. Tests have been run on a Mac I7 and algorithms were implemented in Matlab. We realise there is a huge difference between the three methods. Note the other two methods have to compute the distance between all the fingerprints in each iteration.

Chapter 5: Conclusions & Future Work

5.1 Conclusions and Future Work

Although graph edit distance has become an important tool in structural pattern recognition since it allows us to measure the dissimilarity of attributed graphs, it has two main constraints: it requires an adequate definition of the costs of these operations and its computation cost is exponential with regard to the number of nodes.

This thesis has the three main contributions.

In the first part of the thesis, we have presented a method to learn the edit costs based on embedding the ground truth node-to-node mappings into a Euclidean space. This space has the particularity that the border between substitution and deletions is set as a hyper plane defined by the edit cost parameters.

The learning method is limited to the applications that substitution costs are represented as a linear combination of a vector of weights and a vector of local costs, weights are normalized to one and insertion and deletion costs are constant. Moreover, the method assumes that the GED is approximated through a sub-optimal algorithm, in which the local information is represented through stars. Note that the weights and costs deduced through our algorithm do not guarantee to be the optimal ones in an optimal graph-matching algorithm neither another sub-optimal graph-matching algorithm based on another local information different from the star.

Nevertheless, from a practical point of view, our method has three main advantages.

First, it does not have parameters to be tuned such as a regularization term.

Second, it is not necessary to impose initial values.

Third, the Graph edit distance does not need to be computed in the learning process.

The experimental validation has shown us that the learned parameters are the ones that obtain the highest matching accuracy considering the current methods in the literature. Moreover, our method has a runtime comparable to the fastest method.

In the Second part of the thesis, we applied our method to perform point set alignment based on transforming the point sets in attributed graphs and then deducing the distance between attributed graphs by the graph edit distance. Point set alignment is an old problem and several methods have been presented to solve it. Some of them only make use of the information of the points but other ones convert the point set into an attributed graph, through considering the value around the pixel and also the pixel position.

The main difference between our method applied and the methods based on graphs is that we do not specifically introduce any mechanism related to the fact that graphs represent point sets. The novelty is to show that properly tuning the graph edit distance (learning them through an automatic method) is enough to achieve competitive results in some public

databases. We have used the learning algorithm but other ones could be used.

The experimental corroborates our theory, since the classical graph edit distance, in which the weights on the edit costs have been automatically learned, returns competitive results with regard to the algorithms devoted to perform image alignment.

In the third part of the thesis, we have presented a method to learn the fingerprint parameters based on embedding the ground truth minutia-to-minutia mappings into a 2D space. This space has the particularity that the regression line defines the fingerprint parameters.

From a practical point of view, our method has main advantages. That the experimental validation has shown that our method properly learns the fingerprint parameters since the learned parameters are the ones that obtain the highest recognition ratios and minimum Hamming distances. Moreover, our method is pretty fast since the fingerprint matching is not needed to be run and the computational cost is quadratic with respect to the number of minutiae-to-minutiae impositions fingerprint. Usually, this number is very low, therefore, the learning process runtime is almost negligible.

One important drawback of the other methods, if we consider the runtime is not a handicap in the learning process, is the need of an initial value since there is no clue to guess it. In these case, we suppose that imposing the value learned by our method could be a good initial guess. Nevertheless, we have seen that in this case, the algorithm stops iterating

pretty fast and it tends to return a value similar to this initial guess. Another important limitation is the existence of ground truth mappings.

For Future work, we aim to learn graph edit distance without the need of being constants. In this direction, we are defining a method that use multi class neural network, similar to [81] but modifying the learning model by using all classes in the same time in order to increase the efficiency and accuracy. We are planning to extend this idea by using combination techniques to improve the performance.

Annex : Databases

This annex provides a detailed description of the databases used in the Experiments of the thesis.

Databases for learning graph matching

In pattern recognition, benchmarking is a process to measure the quality of the representation of the objects or the quality of the algorithms involved in comparing, classifying or clustering these objects. The objective of benchmarking is to compare the performance of the involved object representations with the pattern recognition algorithms. The IAM graph database repository [56] composed of twelve databases of diverse attributed graphs has been largely cited and used as a benchmark to compare new algorithms with the current state of the art. The aim of this repository is always been to increase the recognition ratio in a classification framework. Nevertheless, in this thesis we needed to train and evaluate not only the classification accuracy but also the hamming distance. For this reason, a basic requirement in the datasets to use is to dispose of a ground-truth correspondence between the nodes of the graphs of the same class. For this reason, we added a ground-truth correspondence to the databases described in this section. This groundtruth is independent of the graph matching algorithm and also on their specific parameters, since it has been imposed by a human or

deduced using another technique described above. The databases described in this section can be used either to compare the classification accuracy or assess the hamming distance.

1- General Structure

Databases are composed of N registers of elements $\{G^{P^i}, G^{a^i}, \hat{f}^i, C^i\}$ that have a pairs of graphs and a correspondence. Attributed graphs and need to be defined in the same attribute domain (graph class C^i), but may have different orders. The ground-truth correspondence $\hat{f}^i$ between nodes of G^{P^i} and G^{a^i} may have some nodes of G^{P^i} mapped to nodes of G^{a^i}, and other ones mapped to the null node. Nevertheless, two nodes of cannot be mapped to the same node of. The null node is a mechanism to represent that a node of do not have to be mapped to any node of [55]. Note some nodes of may not have been mapped to any node of through.

2- Databases

2.a LETTERS

The letters graph database consists of a set of 2250 graphs that represent artificially distorted 15 different classes of letters of the Latin alphabet. For each class, a prototype line drawing was manually constructed. These prototype drawings are then converted into prototype graphs by

representing lines by undirected edges and ending points of lines by nodes. Attributes on nodes are only the bi-dimensional position of the junctions and edges do not have attributes. Figure A.1 shows 4 samples of letter A. There are three variants of the database depending on the degree of distortion with respect to the original prototype (adding, deleting and moving nodes and edges), viz. low, medium and high (LETTER-LOW,LETTER-MED, LETTER-HIGH). The ground-truth correspondence between the nodes is well-known, because graphs of each class are generated from an original prototype.

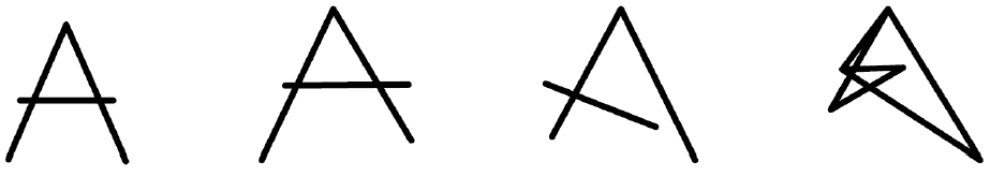


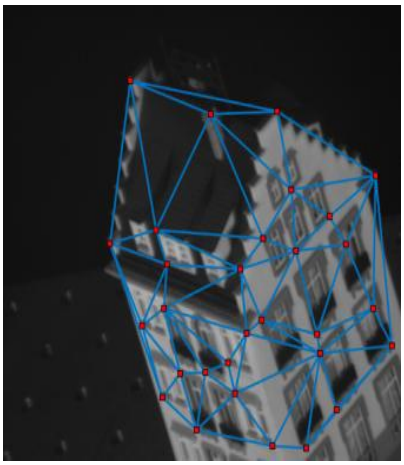
Figure A.1. Different instances of letter A.

2.b HOUSE-HOTEL

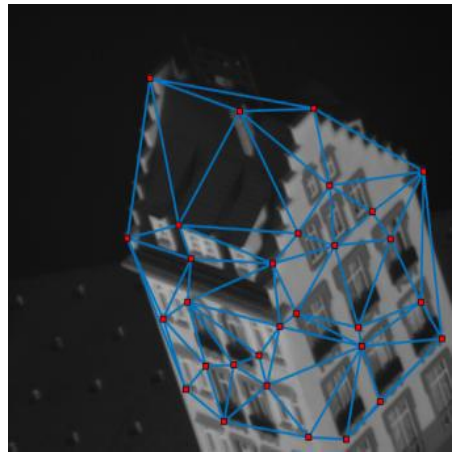
The HOUSE-HOTEL database, consisting in CMU sequences of 111 frames of a toy house and 101 frames of a hotel [147], respectively. Each frame of these sequences has the same 30 hand-marked salient points identified and labelled. Each salient point represents a node on the graph and it has 60 Context Shape features. They triangulated the set of landmarks using the Delaunay triangulation to generate the non-attributed edges of the graph.

They made three sets of pairs of frames considering different baselines (number of frames of separation into the video sequence). One set was used

to learn, another to validate and the third one to test the learned weights. From each set, it has been manually defined the ground-truth correspondence between their graphs.



a. HOUSE



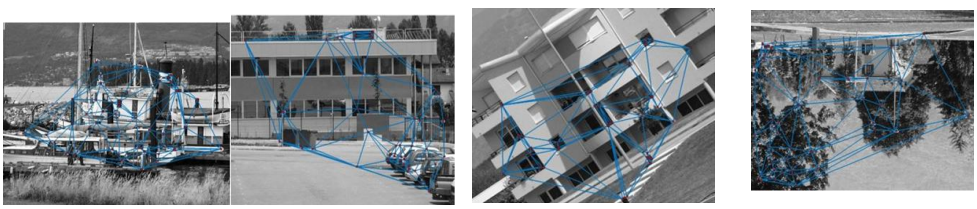
b. HOTEL

Figure A.2. Examples of HOUSE and HOTEL original images databases with salient points and edges.

2.c TARRAGONA ROTATION ZOOM

The public database called Tarragona Rotation Zoom database [217] is composed of 4 sequences. Each sequence has 10 attributed graphs (50 nodes each) and 10 node correspondences. Each node correspondence represents the exact correspondence between the nodes of the first graph in the sequence and the nodes of other graphs. Each attributed graph in eacsequence represents one of the images from the public image databases called BOAT, EAST_PARK, EAST_SOUTH and RESIDENCE. Nodes are salient points from the images extracted through the SURF extractor. Only the best 50 points were selected. Attributes on the nodes are the SURF features.

Edges have been computed through Delaunay triangulation and do not have attributes. Due to the presence of outliers in the salient points, the node correspondences are not bijective. The reference set (test set) of each database consists of 5 attributed graphs extracted from even (odd) images of the initial sequences. The ground-truth correspondence between nodes is deduced from the homography that relates the original images.



a. BOAT b. EAST_PARK c. EAST_SOUTH d. RESID

Figure A.3. The first image of each of the 4 classes and their graphs.

2.d FINGERPRINTS

The FINGERPRINTS database consists of a set of graphs generated from skeletonized fingerprint images. Ending points and bifurcation points of the regions are represented by nodes. Additional nodes are inserted in regular intervals between ending points and bifurcation points. The edges link nodes that are directly connected. Each node is featured with a two dimension AL attribute giving its position. The edges are attributed with an angle denoting the orientation of the edge with respect to the horizontal direction.

Figure A.4. Original examples of the four fingerprint classes.

List of Publications

Journals

Shaima Algabli,Francesc Serratosa ”Embedding the node-to-node mappings to learn the Graph edit distance parameters” Pattern Recognition Letters Volume 112, September 2018, Pages 353-360 (**Quartile Q1.)**

Conferences

- Santacruz, Pep, Shaima Algabli,and Francesc Serratosa” Node matching between two large graphs in linear computational cost” in Pattern Recognition, GbRPR 2017; Anacapri; Italy; 16 May 2017 through 18 May 2017; Code 191749

- Shaima Algabli, Pep Santacruz and Francesc Serratosa” Learning the Graph edit distance parameters for point-set image registration”, CAIP 2019(**Core B**)

- Shaima Algabli, and Francesc Serratosa, “Learning Graph Matching Substitution Weights based on a Linear Regression,”ICPR 2020 **(Core B)** (**under review**)

Workshops

- Shaima Algabli and Francesc Serratosa”Learning the Insertion and Deletion Edit Costs for Graph Matching” 1st URV Doctoral Workshop in Computer Science and Mathematics, 77, 2017.

Bibliography

1. K. Driessens, J. Ramon, T. Gärtner, "Graph kernels and Gaussian processes for relational reinforcement learning." Machine Learning, pp. 64(1-3): 91-119, 2006.

2. P. Mahé, J.-P. Vert, "Graph kernels based on tree patterns for molecules," Machine Learning, pp. 75(1):3-35,, 2009.3.

3. X. Qi, Q. Wu, Y. Zhang, E. Fuller, C.-Q. Zhang., "A novel model for DNA sequence similarity analysis based on graph theory," Evolutionary Bioinformatics (7), pp. 149-15, 2011.

4. D. J. Cook, L. B. Holder, "Mining Graph Data, Wiley, 2006.

5. D. Conte, P. Foggia, C. Sansone, M. Vento, "“Thirty Years Of Graph Matching In Pattern Recognition," Vols. Vol. 18, No. 3, pp. 265-298, 2004.

6. Gao X, Xiao B, Tao D, Li X (2010) A survey of graph edit distance. Pattern Anal Appl 13(1):113–129. https://doi.org/10.1007/s10044-008-0141-y

7. Myers R, Wilson RC, Hancock ER (2000) Bayesian graph edit distance. IEEE Trans Pattern Anal Mach Intell 22(6):628–635. https://doi.org/10.1109/34.862201

8. Riesen K (2015) Structural pattern recognition with graph edit distance. Advances in computer visionand pattern recognition. Springer, Cham https://doi.org/10.1007/978-3-319-27252-8.

9. Serratosa F, Cortés X (2015) Graph edit distance: moving from global to local structure to solve the graph-matching problem. Pattern Recognit Lett 65:204–210

10. Solé-Ribalta A, Serratosa F, SanfeliuA(2012) On the graph edit distance cost: properties and applications.IJPRAI 26(5):1260004

11. Gallagher, B. (2006). Matching structure and semantics: A survey on graph-based pattern matching. In Capturing and Using Patterns for Evidence Detection, Papers from the 2006 AAAI Fall Symposium, Washington,DC, USA, October 13-15, 2006., pages 45–53.

12. Foggia, P., Percannella, G., and Vento, M. (2014). Graph matching and learning in pattern recognition in the last 10 years. International Journal of Pattern Recognition and Artificial Intelligence, 28(01):1450001.

13. S. Kosinov, T. Caelli, "Inexact multisubgraph matching using graph eigenspace and clustering models," Structural, Syntactic, and Statistical Pattern Recognition, p. 133–142, 2002.

14. R.C. Wilson, E.R. Hancock, and B. Luo, "Pattern vectors from algebraic graph theory," IEEE Trans.on Pattern Analysis and Machine Intelligence, p. 27(7):1112–1124, 2005.

15. H. Qiu, E.R. Hancock, "“Graph matching and clustering using spectral partitions," Pattern Recognition, p. 39(1):22–34, 2006.

16. J. Ramon, T. Gärtner, "Expressivity versus efficiency of graph kernels”," First International Workshop on Mining Graphs, Trees and Sequences, p. 65– 74, 2003.

17. K. Borgwardt, T. Petri, H.-P. Kriegel, S. Vishwanathan, "An efficient sampling scheme for comparison of large graphs," Int. Workshop on Mining and Learning with Graphs, 2007.

18. S.V.N Vishwanathan, K. Borgwardt, N.N. Schraudolph, "Fast computation of graph kernels," 2006.

19. B. Gauzere, L. Brun, D. Villemin, "Two new graph kernels and applications to chemoinformatics Int," 2011.

20. B. Gauzere, P.A. Grenier, L. Brun, D. Villemin., "Treelet kernel incorporating cyclic, stereo and inter pattern information in chemoinformatics," Pattern Recognition, p. 48(2):356–367, 2015.

21. M. Vento, "A One Hour Trip in the World of Graphs,Looking at the Papers of the Last Ten Years," Graph Based Representations in Pattern Recognition, pp. 1-10, 2013.

22. A.Sanfeliu, K. Fu, "“A distance measure between attributed relational graphs for pattern recognition”.," IEEE Trans. on Sys., Man and Cybern, vol. 13, pp. 353- 362, 1983.

23. H. Bunke, G. Allermann, "Inexact graph matching for structural pattern recognition," Pattern Recognition Letters, vol. 1(4), pp. 245-253, 1983.

24. Ferrer M, Serratosa F, Riesen K (2015) Improving bipartite graph matching by assessing the assignmentconfidence. Pattern Recognit Lett 65:29–36.

25. Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107. https://doi.org/10.1109/TSSC.1968.300136.

26. Garey MR, Johnson DS (1990) Computers and intractability; a guide to the theory of NP-completeness.W. H. Freeman & Co., New York.

27. Gold S, Rangarajan A (1996) A graduated assignment algorithm for graph matching. IEEE Trans Pattern Anal Mach Intell 18(4):377–388. https://doi.org/10.1109/34.491619.

28. J. Lladós, E. Martí, J.J. Villanueva, Symbol Recognition by Error-Tolerant Subgraph Matching between Region Adjacency Graphs. Trans. Pattern Anal. Mach. Intell. 23(10): 1137-1143, 2001.

29. Riesen K, Fischer A, Bunke H (2018) On the impact of using utilities rather than costs for graphmatching.Neural Process Lett 48(2):691–707. https://doi.org/10.1007/s11063-017-9739-7.

30. Serratosa F (2014) Fast computation of bipartite graph matching. Pattern Recognit Lett 45:244–250.

31. Serratosa F (2014) Speeding up fast bipartite graph matching through a new cost matrix. Int J Pattern Recognit Artif Intell 29:1550010. https://doi.org/10.1142/S021800141550010X.

32. Serratosa F (2015) Computation of graph edit distance: reasoning about optimality and speed-up. Image Vis Comput 40:38–48.

33. Cortés X, Serratosa F, Riesen K (2016) On the relevance of local neighbourhoods for greedy graph edit distance. In: S+SSPR, lecture notes in computer science, vol 10029, pp 121–131.

34. Riesen K, Ferrer M, Dornberger R, Bunke H (2015) Greedy graph edit distance. In: Proceedings of the 11th international conference on machine learning and data mining in pattern recognition—vol 9166, MLDM 2015. Springer, Berlin, pp 3–16. https://doi.org/10.1007/978-3-319-21024-7_1.

35. Abu-Aisheh Z, Raveaux R, Ramel J (2016) Anytime graph matching. Pattern Recognit Lett 84:215–224.

36. Bougleux S, Brun L, Carletti V, Foggia P, Gaüzère B, Vento M (2017) Graph edit distance as a quadratic assignment problem. Pattern Recognit Lett 87:38–46.

37. Santacruz P, Serratosa F (2018) Error-tolerant graph matching in linear computational cost using an initial small partial matching. Pattern Recognit Lett. https://doi.org/10.1016/j.patrec.2018.04.003.

38. X.Cortés, F. Serratosa, "Interactive Graph-Matching using Active Query Strategies," Pattern Recognition, vol. 48(4), pp. 1360-1369, 2015.

39. F.Serratosa, "Fast Computation of Bipartite Graph Matching," Pattern Recognition Letters 45, pp. 244 - 250, 2014.

40. K. Riesen, H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," Image and Vision Computing, Vols. 27,no. 4,, p. 950–959, 2009.

41. K. Riesen, A. Fischer, H. Bunke, "Estimating Graph Edit Distance Using Lower and Upper Bounds of Bipartite Approximations," vol. 29(2), 2015.

42. K. Riesen, H. Bunke, "Improving bipartite graph edit distance approximation using various search strategies," Pattern Recognition, vol. 48(4), pp. 1349-1363, 2015.

43. Harold W. Kuhn, "The Hungarian Method for the assignment problem," Naval Research Logistics Quarterly , p. 2:83–97, 1955.

44. Shaima Algabli and Francesc Serratosa. Embedding the node-to-node mappings to learn the graph edit distance parameters. Pattern Recognition Letters, 112:353–360, 2018.

45. Xavier Cort´es, Donatello Conte, and Hubert Cardot. Learning edit cost estimation models for graph edit distance. Pattern Recognition Letters, 125:256–263, 2019. 3.

46. Michel Neuhaus and Horst Bunke. Self-organizing maps for learning the edit costs in graph matching. IEEE Transactionson Systems, Man, and Cybernetics, Part B (Cybernetics),35(3):503–514, 2005. 3, 6, 7, 8.

47. Michel Neuhaus and Horst Bunke. Automatic learning ofcost functions for graph edit distance. Information Sciences,177(1):239–247, 2007. 3, 7,

48. Tib´erio S Caetano, Julian J McAuley, Li Cheng, Quoc V Le, and Alex J Smola. Learning graph matching. IEEE transactions on pattern analysis and machine intelligence,31(6):1048–1058, 2009. 3, 4, 7.

49. Marius Leordeanu, Rahul Sukthankar, and Martial Hebert. Unsupervised learning for graph matching. International journal of computer vision, 96(1):28–45, 2012. 3, 7.

50. Xavier Cort´es and Francesc Serratosa. Learning graph matching edit-costs based on the optimality of the oracle's node correspondences. Pattern Recognition Letters, 56:22– 29, 2015. 3, 4, 7, 8.

51. Xavier Cort´es and Francesc Serratosa. Cooperative pose estimation of a fleet of robots based on interactive points alignment. Expert Systems with Applications, 45:150–160, 2016.

52. Xavier Cort´es, Donatello Conte, Hubert Cardot, and Francesc Serratosa. A deep neural network architecture to estimate node assignment costs for the graph edit distance.pages 326–336, 2018.

53. Pep Santacruz and Francesc Serratosa. Learning the suboptimal graph edit distance edit costs based on an embedded model. In S+SSPR, volume 11004 of Lecture Notes in Computer Science, pages 282 292. Springer, 2018.

54. Maxime Martineau, Romain Raveaux, Donatello Conte, and Gilles Venturini. Learning error-correcting graph matching with a multiclass neural network. Pattern Recognition Letters, 2018.

55. A. Solé, F. Serratosa, A. Sanfeliu. "On the Graph Edit Distance cost: Properties and Applications". International Journal of Pattern Recognition and Artificial Intelligence, 26(5), pp:1-21, 2012

56. C. Moreno , X. Cortés , F. Serratosa , A graph repository for learning error-tolerant graph matching, in: Syntactic and Structural Pattern Recognition, LNCS 10029, Merida, Mexico, 2016, pp. 519–529 .

57. http://deim.urv.cat/ ~francesc.serratosa/SW/.

58. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cam- bridge University Press, ISBN: 0521540518, second edn. (2004).

59. L.Dryden, I., V.Mardia, K.: Statistical Shape Analysis, Wiley Series in Proba- bility and Statistics: Probability and Statistics. John Wiley & Sons, Ltd., ISBN: 9780470699621 (1998).

60. Kendall, D.: Shape manifolds, procrustean metrics, and complex projective spaces. Lond Math Soc p. 16.81121 (1984).

61. Berge, J.M.F.t.: The rigid orthogonal procrustes rotation problem. Psychometrika71(1),201{205(Mar2006).https://doi.org/10.1007/s11336 -004-1160-5,

62. Umeyama.S: Least squares estimation of transformation parameters between two point patterns. IEEE Trans.Pattern Anal Machine Intell. 13 p. 376380 (1991).

63. Haralick, R.M., Joo, H., Lee, C., Zhuang, X., Vaidya, V.G., Kim, M.B.: Pose es- timation from corresponding point data. IEEE Trans. Systems, Man, and Cyber- netics 19(6), 1426{1446 (1989).

64. Ho, J., Y.M.H.: On a ne registration of planar point sets using complex numbers . Computer Vision and Image Understanding 115 (2011).

65. Dempster, A.P., L.N.R.D.: Maximum likelihood from incomplete data via the em algorithm . Statist.Soc (1977).

66. Gold, S., R.A.: A graduated assignment algorithm for graph matching . IEEE Trans. Pattern Anal. Machine Intell (1996).

67. Gold, S., R.A.L.C.P.P.S.M.E.: New algorithms for 2d and 3d point matching: pose estimation and correspondence. PatternRecognit (1998).

68. Rangarajan, A., C.H.B.F.: The softassign procrustes matching algorithm . Proceedings of the 15th International Conference on Information Processing in Medical Imaging. Springer-Verlag (1997).

69. Myronenko, A., S.X.: Point set registration coherent point drift. IEEE Trans.Pattern Anal. Machine Intell (2010).

70. Chui, H., R.A.: A new algorithm for non-rigid point matching. CVPR (2000).

71. Gill, F.: Computation of graph edit distance: Reasoning about optimality and speed-up. Image Vision Computing 40, 38{48 (2015).

72. Cross, A., H.E.: Graph matching with a dual-step em algorithm. . IEEE Trans. Pattern Anal. Machine Intell (1998).

73. Luo, B., H.E.: A unied framework for alignment and correspondence. Computer Vision and Image Understanding (2003).

74. Sanroma, G., Renzar, Serratosa, F., Herrera, B.: Smooth point-set registration us- ing neighboring constraints. Pattern Recognition Letters 33(15), 2029{2037 (2012).

75. R. Jonker & T. Volgenant, Improving the Hungarian Assignment Algorithm, Operations Research Letters, 5(4), pp: 171-175, 1986.

76. I.B. Vapnyarskii,2001,Lagrange multipliers, Encyclopedia of Mathematics, Springer Science+Business Media B.V. / Kluwer Academic Publishers,ISBN 978-1-55608-010-4

77. P. E. Gill, W. Murray & M. H. Wright. Practical Optimization, Academic Press, London, UK, 1981.

78. D. Dasgupta, R. Arunava, A. Nag, Advances in User Authentication, ISBN 978-3-319-58808-7, 2017.

79. D. Maltoni, D. Maio, A. Jain, S. Prabhakar, Handbook of Fingerprint Recognition, Springer, ISBN: 978-1-84882-254-2, 2009.

80. X. Cortés & F. Serratosa, Learning Graph Matching Substitution Weights based on the Ground Truth Node Correspondence, International Journal of Pattern Recognition and Artificial Intelligence, 30(2), pp: 1650005 [22 pages], 2016.

81. Pep Santacruz and Francesc Serratosa. Learning the graph edit costs based on a learning model applied to sub-optimal graph matching. Neural Processing Letters, pages 1–24, 2019. 3, 4, 5, 7.

82. .

UNIVERSITAT ROVIRA I VIRGILI
LEARNING THE GRAPH EDIT DISTANCE THROUGH EMBEDDING THE GRAPH MATCHING
Shaima Ahmed Algabli

UNIVERSITAT ROVIRA I VIRGILI
LEARNING THE GRAPH EDIT DISTANCE THROUGH EMBEDDING THE GRAPH MATCHING
Shaima Ahmed Algabli

UNIVERSITAT
ROVIRA i VIRGILI