ACTIVE AND INTERACTIVE LEARNING STRATEGIES FOR ERROR-TOLERANT GRAPH MATCHING

Xavier Cortés Llosa

Xavier Cortés Llosa

ACTIVE AND INTERACTIVE LEARNING STRATEGIES FOR ERROR-TOLERANT GRAPH MATCHING

DOCTORAL THESIS

Supervised by Dr. Francesc Serratosa Casanelles

Department of Computer Sciences and Mathematics

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2016

UNIVERSITAT
ROVIRA I VIRGILI

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

I STATE that the present study International Thesis entitled: “Active and interactive learning strategies for Error-Tolerant Graph Matching”, presented by the PhD candidate **Xavier Cortés Llosa**, 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
May 10th, 2016

Thesis Supervisor:

[illegible]

Dr. Francesc Serratosa Casanelles

Agraïments

M'agradaria dedicar les primeres línies d'aquesta tesi a totes aquelles persones que amb el seu suport m'han ajudat a fer-la possible. En primer lloc voldria recordar el Dr. Francesc Serratosa, qui amb el seu coneixement, les seves idees, les seves ganes de treballar i la confiança dipositada en mi, han fet que aquesta tesi tirés endavant amb uns resultats dels que només en puc estar orgullós. També vull recordar al meu company de laboratori, en Carlos, qui ha estat el meu company de viatge durant aquests anys i amb qui junts hem anat superant les dificultats que han pogut sorgir amb esperit cooperatiu i formant un bon equip. Així mateix, vull agrair al Dr. Kaspar Riesen i a tots els companys de la FHNW per la seva hospitalitat durant la meva estada i les seves aportacions al treball que aquí es presenta. Tampoc em puc deixar de recordar a la meva família, especialment els meus pares i la meva germana, que tot i no comprendre del tot el que estava fent, sempre m'han donat tot el suport que estava a les seves mans. Finalment, vull recordar a la meva companya Eliana, qui ha estat sempre al meu costat i ha alleugerit la càrrega en moments difícils amb la seva companyia, comprensió i amor.

Tarragona, maig de 2016

Resum

Els grafs, són un tipus de dades que ens permet emmagatzemar la informació estructural d'un objecte conferint-nos la possibilitat de representar patrons que degut a la seva pròpia naturalesa requereixen d'aquesta particularitat, ja siguin imatges, estructures químiques o biològiques, xarxes, patrons biomètrics...

Des de fa més de 30 anys, la recerca enfocada a com representar objectes mitjançant grafs i el posterior còmput de la distància entre aquestes representacions, ha ocupat el treball de molts investigadors. La definició d'un model adequat per mesurar la dissimilitud entre dues d'aquestes representacions, és una qüestió clau en el camp del reconeixement de patrons. A aquest problema se l'ha anomenat *Error-Tolerant GraphMatching*. Una de les aproximacions més populars per tal de trobar-hi una solució ha estat la distància d'edició entre grafs (*Graph Edit Distance*), que consisteix a estimar la distància a partir de la suma del cost requerit per una sèrie d'operacions d'edició que transformen el graf objectiu amb el que estem intentant comparar.

A la primera part d'aquesta tesi es proposen diferents mètriques per calcular la dissimilitud entre dues subestructures locals corresponents a dos dels nodes dels grafs que estem comparant i es demostra com cadascuna d'aquestes mètriques afecta a la distància estimada resultant, en termes de precisió i temps de còmput requerit.

Posteriorment, es defineixen una sèrie d'estratègies actives d'aprenentatge per tal d'afegir interactivitat al problema. Aquestes estratègies proposen a l'usuari, d'un en un, els aparellaments més dubtosos entre nodes de la solució trobada automàticament per tal de que hi faci les correccions oportunes, així, i després d'algunes interaccions el sistema acaba trobant la correspondència entre els grafs perfecta d'acord amb el criteri de l'usuari.

Per altra banda, el problema de la distància d'edició requereix que siguin definits un conjunt de paràmetres per assignar un cost a cadascuna de les operacions d'edició i poder calcular la distància final. La forma més habitual de parametrització ha estat a través d'un procés manual de prova i error sobre cada domini de dades amb el que es treballa. En aquesta tesi es proposa un model per tal d'aprendre aquests costos de forma automàtica a partir d'un conjunt de parelles de grafs i la correspondència entre els seus nodes, tal que les correspondències automàtiques entre els grafs calculades amb els paràmetres apresos s'aproximin a les desitjades.

Deixant el problema de la distància d'edició de grafs, en aquesta tesi també es presenta un model basat amb metric-trees de prototips de classes de grafs per emmagatzemar col·leccions d' aquests. Utilitzant la ja coneguda estratègia d'agrupar les dades que anem guardant de forma intel·ligent per no haver d'explorar totes les instàncies que tenim quan volem fer una cerca, però afegint el concepte de prototip com a estructura intermèdia.

Finalment, proposem portar el concepte d'interactivitat a un altre domini, el de la relació de punts entre dues imatges, per poder millorar la precisió en el càlcul de la posició correlativa entre diferents robots que pertanyen a una mateixa flota que treballa de forma cooperativa.

Abstract

Graphs are data types that can store structural information of objects and are commonly used to represent patterns that because of its nature require this peculiarity, as images, chemical or biological structures, networks, biometric patterns...

For 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 for measure the dissimilarity between these representations is a key issue in pattern recognition. This is the *Error-Tolerant Graph Matching* problem. 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 graph computing the sum of cost of different edit operations that transform one graph into another.

The first part of this thesis presents different metrics to estimate dissimilarity between local substructures of two nodes. We demonstrate how metrics directly affect the performance of *Graph Edit Distance* in terms of computational cost and classification accuracy.

Next, we define some active learning strategies adding interactivity to the *Error-Tolerant Graph Matching*. These strategies propose the most uncertain mappings between nodes automatically found in a complete correspondence set between two graphs and the user imposes corrections (or not) over this mappings. We demonstrate how after some interactions the system finds the perfect graphs correspondence according to the user criteria.

On the other hand, *Graph Edit Distance* model requires configuring a set of parameters to assign an appropriate penalization cost to each edit operation. The most common way to fix these parameters is through a manual process of trial and error over each data domain. We propose a new model to learn these costs automatically.

This thesis also presents a model based on metric-trees of *Graph-Class Prototypes* to store large collections of graphs. The proposed model is based on smartly grouping the data in a database.

Finally, we propose to bring the interactivity to a different domain, the problem of matching points between two images in order to improve the accuracy calculating the relative position between different robots of a fleet working cooperatively.

Contents

Chapter 1 Introduction

1.1 Graphs and Pattern Recognition

Graphs are defined by a set of individual components (nodes) and structural relations between them (edges)[1], offering a convenient way to formally model objects or patterns which are composed of complex subparts including the relations that might exist between these subparts. *Attributed graphs* are graphs in which some attributes are added on nodes and edges to represent local information or characterisation, becoming more relevant in a variety of applications, including 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 representation of 2D or 3D objects, handwritten characters, networks, proteins, fingerprints, and so on [5]. Note that, to generate graphs, the pattern recognition process has to extract them from the objects. This is not a trivial task since the quality of graph based representation is crucial for the rest of the process, but this will not be the central aim of the thesis.

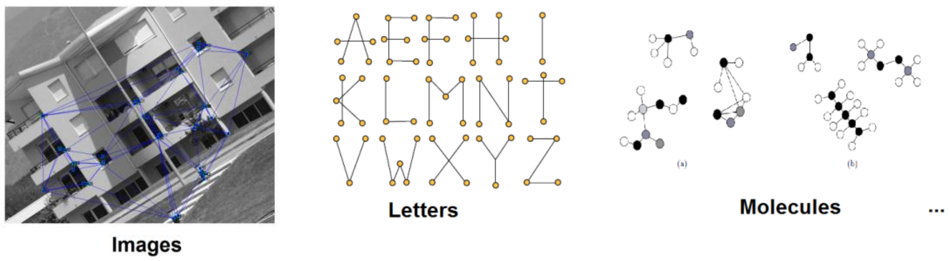


Figure 1.1. Attributed graphs can represent the structural relations between the component parts of the objects.

1.2 Attributed Graphs

We formally define an *attributed graph* as a triplet $G = (\Sigma_v, \Sigma_e, \gamma_v)$, where $\Sigma_v = \{v_i \mid i = 1, \dots, n\}$ is the set of nodes and $\Sigma_e = \{e_{ij} \mid i, j \in 1, \dots, n\}$ is the

[1] The terms nodes/vertices and arcs/edges are used indistinctly in the literature. In this thesis will always be referred to as nodes and edges.

set of undirected and unattributed edges[2]. Function $\gamma_v : \Sigma_v \to \Psi^t$ assigns attribute values from an arbitrary domain (defined in a metric space of $t \in \mathbb{N}$ dimentions) to nodes. The order of a graph G is equal to the number of nodes n. The number of edges of node v_i is referred to as $E(v_i)$. Finally, the local structure[3] of a node v_i is generically referred as N_{v_i} (in Chapter 2 we discuss about different local structures).

1.2.1 Correspondences between graphs

We define a correspondence between two graphs or simply a correspondence, as follows:

Let $G^p = (\Sigma_v^p, \Sigma_e^p, \gamma_v^p)$ and $G^q = (\Sigma_v^q, \Sigma_e^q, \gamma_v^q)$ be two attributed graphs of initial order n and m. 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 $\hat{\Sigma}_v^p \subseteq \Sigma_v^p$ and $\hat{\Sigma}_v^q \subseteq \Sigma_v^q$ respectively. Let T be a set of all possible correspondences between two sets Σ_v^p and $\hat{\Sigma}_v^q$. We define the non-existent or null edges by $\hat{\Sigma}_e^p \subseteq \Sigma_e^p$ and $\hat{\Sigma}_e^q \subseteq \Sigma_e^q$. Correspondence $f^{p\to q} : \Sigma_v^p \to \Sigma_v^q$, assigns bijectively one node of G^p to only one node of G^q. The correspondence between edges is implicitly defined accordingly to the correspondence of their terminal nodes.

1.2.1.1 Correspondences matrix

A correspondence f can be represented mathematically by means of a matrix $F \in \left((n+m) X\, (n+m)\right)$ where $F_{ia} \in \{0,1\}$ such that $F_{ia} = 1$ if $v_i^p \to v_a^q$ and $F_{ia} = 0$ otherwise. Due to the mapping has to be bijective, we have that $\sum_{i=1}^{n+m} F_{ia} = 1$ and $\sum_{a=1}^{n+m} F_{ia} = 1$.

1.2.1.2 Normalised hamming distance between graphs correspondences

In order to compare two correspondences between nodes, the *hamming distance* (Δ^H) represents how far a particular correspondence f' is with respect to another one f''.

[2] In the present thesis we investigate graphs with attributed nodes only. Yet, all of our concepts can be extended to graphs with attributed edges.

[3] "Local structure" is also termed "neighbourhood" by some authors.

$$\Delta^{H}(f^{'},f^{''}) = 1 - \frac{1}{n+m}\sum_{ia=1}^{n+m} F_{ia}^{'} \cdot F_{ia}^{''} \qquad (1.1)$$

1.3 Error-Tolerant 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 [6, 7, 8, 9, 10]. *Graph kernels* constitute another important family of *Error-Tolerant Graph Matching* procedures and various types of *graph kernels* emerged during the last decade [11, 12, 13, 14, 15]. For an extensive review on these and other *Error-Tolerant Graph Matching* methods developed during the last forty years, the reader is referred to [16, 17, 5], but probably the most well known model for *Error-Tolerant Graph Matching* is the *Graph Edit Distance*.

1.3.1 Graph Edit Distance

The *Graph Edit Distance* dissimilarity model [18, 19] defines a distance between two graphs G^p and G^q by means of the minimum amount of distortion required to transform G^p and G^q. To this end, a number of distortions or edit operations, consisting of insertion, deletion, and substitution of both nodes and local structures are employed. Edit cost functions are typically introduced to quantitatively evaluate the level of distortion of each individual edit operation. The basic idea of this is to assign a cost to the edit operations proportional to the amount of distortion they introduce in the underlying graphs.

A sequence $(e_1, \dots, e_k)$ of k edit operations that transform G^p completely into G^q is called *edit path* $\lambda(G^p \rightarrow G^q)$ between G^p and G^q. Note that in *edit path* $\lambda(G^p \rightarrow G^q)$ each node of G^p is either deleted or uniquely substituted with a node in G^q, and likewise, each node in G^p is either inserted or matched with a unique node in G^q. The same applies for the local structures.

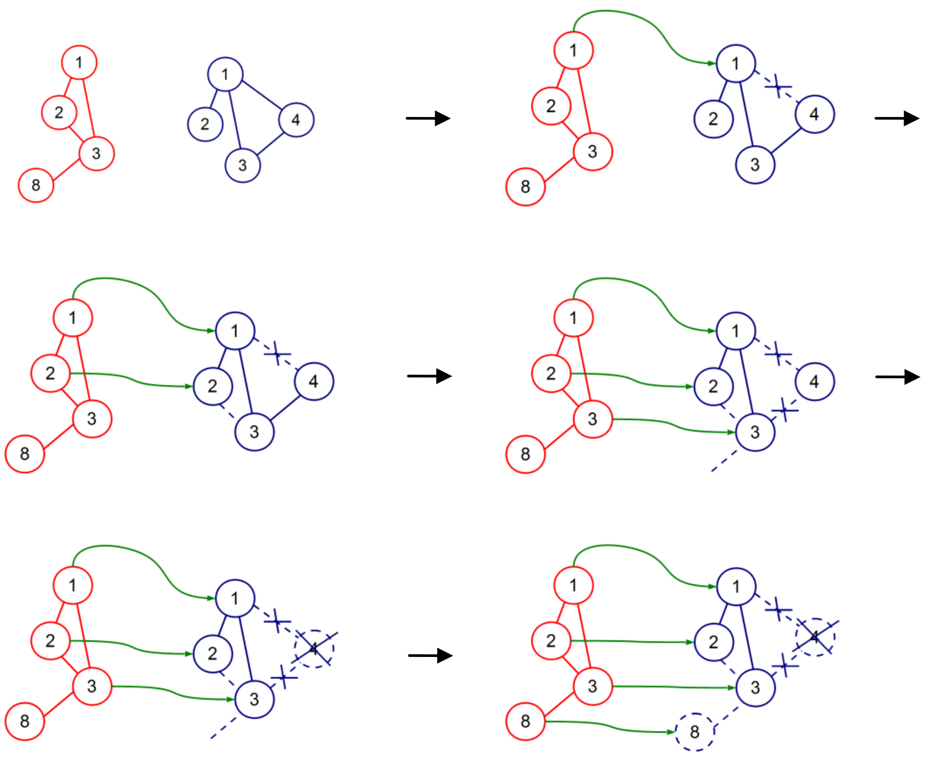


Figure 1.2. An example of an edit path between two graphs. Green arrows: substitutions. Studded: deletions. Dotted: insertions.

Note that given a local structure is inseparable from its node, an *edit path* $\lambda(G^p \to G^q)$ can be related to an univocal correspondence f between nodes of the graphs.

Deletion/insertion operations are transformed to assignations in f of non-null nodes of the first/second graph to null nodes of the second/first graph. Substitutions simply indicate node-to-node assignations in f. Using this transformation, given two graphs, G^p and G^q, and a correspondence between their nodes, f, the graph edit cost is given by:

$$EditCost(G^p, G^q, f) = \qquad \textbf{(1.2)}$$

$$\sum_{\substack{v_i^p \in \Sigma_v^p - \hat{\Sigma}_v^p \\ v_a^q \in \Sigma_v^q - \hat{\Sigma}_v^q}} C_{vs}(v_i^p, v_a^q) + \sum_{\substack{v_i^p \in \Sigma_v^p - \hat{\Sigma}_v^p \\ v_a^q \in \hat{\Sigma}_v^q}} C_{vd}(v_i^p, v_a^q) +$$

$$\sum_{\substack{v_i^p \in \hat{\Sigma}_v^p \\ v_a^q \in \Sigma_v^q - \hat{\Sigma}_v^q}} C_{vi}(v_i^p, v_a^q) + \sum_{\substack{e_{ij}^p \in \Sigma_e^p - \hat{\Sigma}_e^p \\ e_{ab}^q \in \Sigma_e^q - \hat{\Sigma}_e^q}} C_{es}\left(e_{ij}^p, e_{ab}^q\right) +$$

$$\sum_{\substack{e_{ij}^p \in \Sigma_e^p - \hat{\Sigma}_e^p \\ e_{ab}^q \in \hat{\Sigma}_e^q}} C_{ed}\left(e_{ij}^p, e_{ab}^q\right) + \sum_{\substack{e_{ij}^p \in \hat{\Sigma}_e^p \\ e_{ab}^q \in \Sigma_e^q - \hat{\Sigma}_e^q}} C_{ei}\left(e_{ij}^p, e_{ab}^q\right)$$

Being $f(v_i^p) = v_a^q$ and $f\left(v_j^p\right) = v_b^q$

where C_{vs} is the cost of substituting node v_i^p of G^p by node v_a^q of G^q, C_{vd} is the cost of deleting node v_i^p of G^p and C_{vi} is the cost of inserting node v_a^q of G^q. Equivalently for edges, C_{es} is the cost of substituting edge e_{ij}^p of graph G^p by edge e_{ab}^q of G^q, C_{ed} is the cost of assigning edge e_{ij}^p of G^p to a non-existing edge of G^q and C_{ei} is the cost of assigning edge e_{ab}^q of G^q to a non-existing edge of G^p. The cost of mapping two null nodes or null edges is always zero. With these costs, the *Graph Edit Distance* (GED) is defined as the minimum cost under all possible correspondences T.

$$GED(G^p, G^q) = \min_{f \in T} EditCost(G^p, G^q, f) \qquad \textbf{(1.3)}$$

Finally, the optimal correspondence $\dot{f}$ is the one that obtains the minimum cost,

$$\dot{f} = \operatorname*{argmin}_{f \in T} EditCost(G^p, G^q, f) \qquad \textbf{(1.4)}$$

Using these definitions, the *Graph Edit Distance* depends on C_{vs}, C_{vd}, C_{vi}, C_{es}, C_{ed} and C_{ei} costs and several definitions of these costs have been published. There are two main options to define the costs on substituting nodes and edges, C_{vs} and C_{es}. The first one considers cost $C_{vs} \in \{0, K_{vs}\}$ where $C_{vs}(v_i^p, v_a^q) = K_{vs}$ if $dist(v_i^p, v_a^q) >$ Threshold otherwise $C_{vs} = 0$ (similarly for C_{es}). Function $dist$ is defined as a distance over the domain of the attributes. Specific examples of this cost can be found in [20, 21]. The second one corresponds to the case where $C_{vs}(v_i^p, v_a^q) \in \mathbb{R}$ or $C_{es}(e_{ij}^p, e_{ab}^q) \in \mathbb{R}$. In this case, node and edge substitution costs depend on the attributes of the nodes and edges that can be weighed as shown in [39] among others. Functions C_{vd}, C_{vi}, C_{ed} and C_{ei}, are usually defined as a constant, but in some cases depend on node or edge attributes [22, 23, 24].

1.3.2 Sub-optimal Error-Tolerant Graph Matching Computation

Optimal algorithms for computing the *Graph Edit Distance* are typically based on combinatorial search procedures (such as A^* based search techniques). These procedures explore the space of all possible mappings of the nodes and edges of G^p to the nodes and edges of G^q (*i.e.* the search space corresponds to the set of all edit paths $\lambda(G^p \to G^q)$). Yet, considering n nodes in G^p and m nodes in G^q, the set of possible *edit paths* $\lambda(G^p \to G^q)$ contains $O(n^m)$ *edit paths*. Therefore, exact *Graph Edit Distance* computation is exponential in the number of nodes of the involved graphs.

1.3.2.1 Bipartite Graph Matching

The problem of minimizing the *Graph Edit Distance* can be reformulated as an instance of a *Quadratic Assignment Problem* (QAP). QAPs belong to the most difficult combinatorial optimization problems for which only exponential run time algorithms are known to date (QAPs are known to be NP-complete). The *Bipartite Graph Matching* algorithm (BP-GED) [25] is an approximation for the *Graph Edit Distance* that reduces the QAP of graph edit distance computation to an instance of a *Linear Sum Assignment Problem* (LSAP). This algorithm first generates a cost matrix C which is based on costs of editing local substructures of both graphs. Formally, the cost matrix is defined by:

$$C = \left[\begin{array}{cccc|cccc} C_{1,1} & C_{1,2} & \dots & C_{1,m} & C_{1,\varepsilon} & \infty & \dots & \infty \\ C_{2,1} & C_{2,2} & \dots & C_{2,m} & \infty & C_{2,\varepsilon} & \dots & \infty \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{n,1} & C_{n,2} & \cdots & C_{n,m} & \infty & \infty & \dots & C_{n,\varepsilon} \\ \hline C_{\varepsilon,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & C_{\varepsilon,2} & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \cdots & C_{\varepsilon,m} & 0 & 0 & \cdots & 0 \end{array}\right]$$

Where we define $C_{i,\mathrm{a}}$, $C_{i,\varepsilon}$,and $C_{\varepsilon,a}$ according to the following three cases:

1) If the original nodes $v_i^p \in \Sigma_v^p$ and $v_a^q \in \Sigma_v^q$ are mapped then

$$C_{i,\mathrm{a}} = \beta \cdot C_{vs} + (1-\beta) \cdot C_{es} \tag{1.5}$$

where $\beta \in]0,1[$ is a weighting parameter that controls what is more important, the cost of the pure node substitution $C_{vs} = C(v_i^p \to v_a^q)$ or the cost of substituting the local structures $C_{es} = C(N_{v_i^p} \to N_{v_a^q})$ of both nodes.

$$C_{i,\mathrm{a}} = \beta \cdot C(v_i^p \to v_a^q) + (1-\beta) \cdot C(N_{v_i^p} \to N_{v_a^q}) \tag{1.6}$$

2) If one node $v_i^p \in \Sigma_v^p$ in G^p is deleted, we have

$$C_{i,\varepsilon} = \beta \cdot K_v + (1-\beta) \cdot C_{ed} \tag{1.7}$$

where $\beta \in]0,1[$ (as defined above), K_v refers to a positive constant cost for deleting one node and $C_{ed} = C(N_{v_i^p} \to \varepsilon)$ refers to the cost of deleting the complete local structure of v_i^p.

$$C_{i,\varepsilon} = \beta \cdot K_v + (1-\beta) \cdot C(N_{v_i^p} \to \varepsilon) \tag{1.8}$$

3) If one node $v_a^q \in \Sigma_v^q$ in G^q is inserted, we finally have (similar to case 2)

$$C_{i,\varepsilon} = \beta \cdot K_v + (1-\beta) \cdot C_{ei} \tag{1.9}$$

where $C_{ei} = C(\varepsilon \to N_{v_a^q})$

$$C_{\varepsilon,a} = \beta \cdot K_v + (1-\beta) \cdot C(\varepsilon \rightarrow N_{v_a^q}) \qquad \textbf{(1.10)}$$

$C(v_i^p \rightarrow v_a^q)$ is a distance function defined through the node attribute values and K_v gauges the importance of deleting or inserting nodes in the matching process. $C_{i,a}$ is the cost to substitute the node and its local sub-structure and $C_{i,\varepsilon}$ and $C_{\varepsilon,a}$ are the costs to delete and insert it, respectively. These costs depend on the used local sub-structures. Finally, the weighting parameter β has to be set in a validation or learning process.

A linear assignment algorithm can be applied on C in order to find a (optimal) correspondence of nodes and their local structures. A large number of solvers for linear sum assignment problems exist [26]. The time complexity of the best performing exact algorithms for LSAPs is cubic in the size of the problem.

Any complete assignment of local substructures derived on C can be reformulated as an admissible *edit path* from $\lambda(G^p \rightarrow G^q)$. That is, the global edge structure from G^p and G^q can be edited with respect to the node operations captured in the mapping of local substructures (this is due to the fact that edit operations on edges always depend on the edit operations actually applied on their adjacent nodes). Eventually, the total cost of all edit operations (applied on both nodes and edges) can be interpreted as a *Graph Edit Distance* approximation between graphs G^p and G^q (termed BP-GED from now on).

The edit path found with this particular procedure considers the structural information of the graphs in an isolated way only (singular structures). Yet, the derived distance considers the local structure of G^p and G^q in a global and consistent way and thus the derived distance is in the best case equal to, or in general larger than the exact *Graph Edit Distance*.

Recently, new approaches which use an approximation rather than an exact algorithm to suboptimally solve the LSAP stated on cost matrix C has been proposed in [27, 28]. Moreover, in [29] it is proposed a new cost matrix and a different solver to fix the assignment problem. The remaining parts of these new approaches are identical with the original *Bipartite Graph Matching* framework. That is, based on the found assignment of local substructures an admissible edit path and its corresponding sum of costs is derived. However, as the complexity of this suboptimal assignment algorithm decrease and the time complexity of the complete *Graph Edit Distance* approximation is further reduced.

1.3.2.2 Probabilistic Approach

Considering that the involved graphs have a degree of disturbance and also the exponential complexity of the problem, there are several *Error-Tolerant Graph Matching* algorithms that return a probability matrix to find the best correspondence f and a distance between two graphs like *Probabilistic Relaxation* [30], *Graduated-Assignment* [31] and *Expectation-Maximisation* [33]. In fact, the input of these algorithms can be matrices C_v and C_e capturing the assignation costs between nodes and edges of both graphs instead of graphs G^p and G^q. We represent this matrix by P where each cell contains $P\left[v_i^p, v_a^q\right] = Prob\left(f\left(v_i^p\right) = v_a^q\right)$. Thus, given the probability matrix P, it is necessary to derive the final correspondence f by a discretization process. There are several techniques to perform this discretization, *e.g.* [69]. Figure 1.3 represents the probabilistic graph matching paradigm.

In general, if we want to solve the *Error-Tolerant Graph Matching* problem based on probabilities [32, 33, 30], given two graphs G^p and G^q, the objective function to optimize corresponds to the quadratic assignment problem objective function,

$$\begin{aligned} &C_P(G^p, G^q) \\ &= \sum_{v_i^p \in \Sigma_v^p} \sum_{v_a^q \in \Sigma_v^q} \sum_{\substack{v_j^p \in \Sigma_v^p \\ v_j^p \neq v_i^p}} \sum_{\substack{v_b^q \in \Sigma_v^q \\ v_b^q \neq v_a^q}} P\left[v_i^p, v_a^q\right] P\left[v_j^p, v_b^q\right] C_e\left[v_i^p, v_a^q, v_j^p, v_b^q\right] \\ &+ \sum_{v_i^p \in \Sigma_v^p} \sum_{v_a^q \in \Sigma_v^q} P\left[v_i^p, v_a^q\right] C_v\left[v_i^p, v_a^q\right] \end{aligned} \qquad (1.11)$$

where P is restricted to

$$\sum_{v_i^p \in \Sigma_v^p} P\left[v_i^p, v_a^q\right] = 1, \forall v_a^q \in \Sigma_v^q \quad \text{and} \quad \sum_{v_a^q \in \Sigma_v^q} P\left[v_i^p, v_a^q\right] = 1, \forall v_i^p \in \Sigma_v^p \qquad (1.12)$$

Some methods use a *Gradient Ascent* technique [34] or a similar technique to get a local maximum of C_P where $P\left[v_i^p, v_a^q\right]$, $\forall v_i^p \in \Sigma_v^p$ and $\forall v_a^q \in \Sigma_v^q$, are the set of variables of the function. This technique takes steps proportional to the magnitude of the positive gradient with the aim of approaching to a local maximum of function C_P. The magnitude of the gradient of C_P with respect to variable $P\left[v_i^p, v_a^q\right]$ is,

$$M\left(v_i^p, v_a^q\right) = \frac{dC_P(G^p, G^q)}{dP\left[v_i^p, v_a^q\right]} = \sum_{\substack{v_j^p \in \Sigma_v^p \\ v_j^p \neq v_i^p}} \sum_{\substack{v_b^q \in \Sigma_v^q \\ v_b^q \neq v_a^q}} P\left[v_j^p, v_b^q\right] C_e\left[v_i^p, v_a^q, v_j^p, v_b^q\right] \quad \textbf{(1.13)}$$

In Chapter 3, we show how an oracle can impose his feedback to update matrices C_v and C_e. Besides, we present different strategies which, with the information of the probability matrix P and the magnitude of the gradient M, derive the node that has to be queried to the oracle.

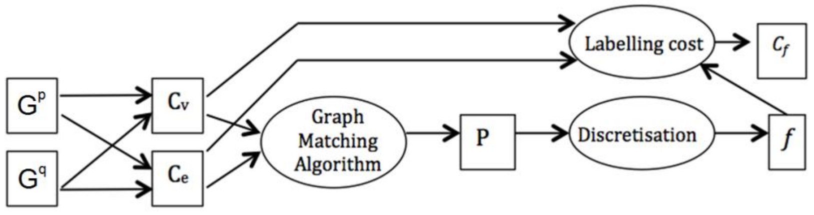


Figure 1.3. Probabilistic graph matching framework.

1.3.3 Learning Error-Tolerant Graph Matching

Methods based on *Bipartite Graph Matching* or on a probabilistic approach for *Error-Tolerant Graph Matching* require assigning a penalty cost to each edit operation according to the amount of distortion that it introduces in the transformation.

An interesting question arises in this context: given the *attributed graphs* that represent some objects, how we gauge the weights of the attributes or the importance of each edit operation? Suppose we have an image retrieval application in which images are represented by adjacency graphs. In these graphs, one of the attributes on the nodes is the average colour hue of the region that the node represents and another attribute is the area of this region. If we consider the average colour hue of the region is more important than the area of the region, the "best" correspondence between nodes of both graphs is going to be different if we consider the area is more important than the average colour hue. Usually, there are some weights in the definitions of the distance between graphs to gauge the level of importance of each attribute [35, 21]. These distance weights are manually set at the validation process and little research has been carried out to automatically set them.

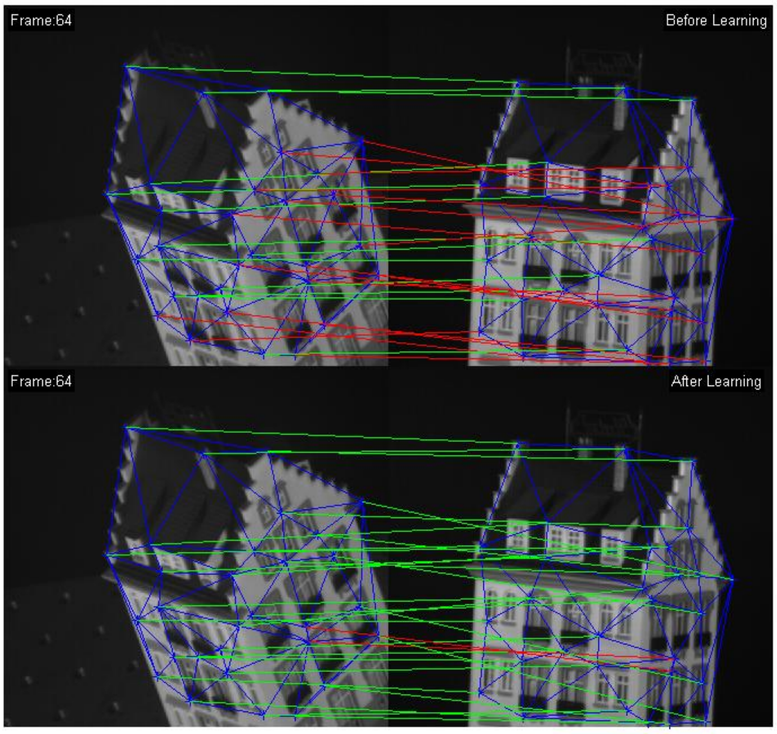


Figure 1.4. An example of correspondences before learning (image above) and after learning (image bellow), between two graphs representing frames of a video sequence. Nodes on graphs are the extracted salient points. Blue lines are the edges between these nodes. Green lines: correct correspondences. Red lines: incorrect correspondences.

Figure 1.4 shows the automatically obtained correspondence before learning the distance weights (above) and after learning (below) these weights. We see the number of correct node mappings (green lines) increases and the number of incorrect node mappings (red lines) decreases after the learning process.

Ref.	Year	Optimisation function	Outlier nodes	Optimisation Algorithm
[37]	2005	-Recognition Ratio	Yes	-Self Organising Maps [41]
[38]	2007	-Recognition Ratio	Yes	-Expectation Minimisation [42]
[39]	2009	-Hamming Distance	No	-Bundle Method [43]
[40]	2012	-Hamming Distance -Recognition Ratio	No	-Spectral [45]

Table 1.1. Papers published about learning graph matching.

Two research directions are related to automatically obtaining the edit costs. The first one approaches to graph classification problem [36] to obtain the edit costs such that classification ratio is maximised [37, 38]. Whereas the aim of the second one is to obtain the edit costs such that the distance between the automatic obtained correspondence and the ground-truth correspondence is minimised [39, 40]. In this second case, we discern between unsupervised and supervised methods. The unsupervised methods are the ones that the ground-truth correspondence is not needed in the learning process [40] and the supervised ones assume a ground-truth correspondence is accessible and they use this knowledge [39]. The first direction methods are applied to pattern recognition or classification whereas the second direction methods can be applied to a more broadly application areas such as simply matching two images or two sets of points. In [37, 38], they propose a method to automatically learn the weights 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 Maximisation* algorithm. In [40], they present an unsupervised method based on the gradient ascent of the first eigenvector of a matrix that represent the pairwise relations between nodes and edges of both graphs. In [39] the authors propose a supervised method in which the optimal predictor to fix the weights that minimize the same general expression (*hamming distance*).

Table 1.1 lists the five papers that have been published related to learning the edit costs. The three main features of these methods are: The optimisation function, outliers and the optimisation algorithm.

1.4 Graph databases

Indexing structures is another fundamental issue in pattern recognition and a key tool in database technology; they are used to obtain efficient access to large collections of objects representations. Traditional image database systems manage global properties of images, such as histograms [46]. Many techniques for indexing one-dimensional data sets have been defined. Since a total order function over a particular attribute domain always exists, this ordering can be used to partition the data and moreover it can be exploited to efficiently support queries. Several multi-dimensional indexes have appeared, such as, colour, texture, shape, with the aim of increasing the efficiency in executing queries on sets of objects characterized by multi-dimensional features.

Effective access to image databases requires queries addressing the expected appearance of searched images [47]. To this end, it is needed to represent the image as a set of entities and relations between them. The effectiveness of retrieval may be improved by registering images as structural elements rather than global features [48, 49]. In the most practiced approach to content-based image retrieval, the visual appearance of each spatial entity is represented independently by a vector of features. Mutual relationships between entities can be taken into account in this retrieval process. Thus, local entities and mutual relationships may be considered to have the same relevance and to be defined as parts of a global structure that captures mutual dependencies [50]. In this case, the model of content takes the structure of an *attributed graph*.

While the distance between two sets of independent features can be computed in polynomial time [32, 51], the exact distance between two graphs is computed in exponential time with respect to the number of nodes of the graphs. Although, as we have been seeing throughout this thesis, some sub-optimal solutions have been presented to compare a pair of graphs, in which, the computational complexity is reduced to polynomial cost.

Out of the specific context of content-based image retrieval, the problem of comparing an input graph against a large number of model graphs has been addressed in several approaches. In some applications, the classes of objects are represented explicitly by a set of graphs, which means that a huge amount of model graphs must be matched with the input graph and so the conventional *Error-Tolerant Graph Matching* algorithms must be applied to each model-input pair sequentially. As a consequence, the total computational cost is linearly dependent on the number of model graphs and exponential (or polynomial if suboptimal methods are used) with the size of the graphs. For applications dealing with large databases, this may be prohibitive. To alleviate these problems, some attempts have been designed with the aim of reducing the computational time of matching the unknown input patterns to the whole set of models from the database. Those approaches assume that the graphs that represent a cluster or class are not completely dissimilar in the database and, in this way, only one structural model is defined from the graphs that represent the cluster. These structures are called *Graph-Class Prototypes*. In the classification process, only one comparison is needed for each cluster.

Considering the current state of the art, some indexing techniques have been developed for graph queries. We divide these techniques into two categories. In the first ones, the index is based on several tables and filters [52, 53]. In the second ones, the index structure is based on metric-trees [54, 55, 56].

In the first group of techniques, the ones that are not based on trees, we emphasize the method developed by *Shasha et. al.* [53] called *GraphGrep*. *GraphGrep* is based on a table in which each row stands for a path inside the graph (up to a threshold length) and each column stands for a graph. Each entry in the table compounds to the number of occurrences of a particular path in the graph. Queries are processed in two phases. The filtering phase generates a set of candidate graphs for which the count of each path is at least that of the query. Since indexing schemes based on paths do not ensure graph isomorphism, in a verification phase, each candidate is strictly compared to the query graph and only isomorphic graphs are returned. On the other hand, Yan *et. al.* [52] proposed G_{Index} that uses frequent patterns as indexing features. These frequent patterns reduce the index space as well as improve the filtering rate. The main drawback of these models is that the construction of the indices requires an exhaustive enumeration of the paths or fragments that increases the memory and time requirements of the model. Moreover, since paths or fragments carry little information about a graph, the lost of information at the filtering step seems to be unavoidable.

Considering the second group, the first time that metric trees were applied to graph databases was done by *Berretti et. al.* [54]. *Attributed graphs* were clustered hierarchically according to their mutual distances and indexed by metric-trees [57]. Queries are processed in a top-down manner by routing the query along the index tree. Each node of the index tree represents a cluster and it has one of the graphs of the cluster as a representative. The graph matching problem, in the tree construction and at query time, was solved by an extension of the *A** algorithm that uses a look-ahead strategy plus a stopping threshold. A drawback of this method is that the computational cost is exponential with respect to the number of nodes in the graphs. Lee *et. al.* [56] used this technique to model graphical representations of foreground and background scenes in videos. The resulting graphs were clustered using the edit-distance metric, and similarity queries were answered using a multi-level index structure.

More recently, *He* and *Singh* [55] proposed what they called a *Closure-tree*. It uses a similar structure than the one presented by *Berretti* [54] but, the representative of the cluster was not one of the graphs but a graph prototype (called closure graph) that could be seen as the union of the *attributed graphs* that compose the cluster. The structurally similar nodes that have different attributes in the graphs are represented in the *Closure graph* with only one node but with more than one attribute. *Closure-trees* have two main drawbacks. First, they can only represent discrete attributes at nodes of the *attributed graphs*. Second, they tend to generalize too much the set of graphs they represent, allowing graphs that have not been used to synthesize the closure graph.

1.5 Objectives and organization of this thesis

This thesis is organized as follows; the thesis is divided in 8 chapters, the present chapter introduces the reader to the domain and the final chapter presents the conclusions. In the middle are discussed the issues and models presented in the thesis.

Chapter 2 is devoted to delve into the definition of the local structure of a node and propose eight different options to compute the distances or dissimilarities between them for *Graph Edit Distance* computation.

Chapter 3, propose a model for perform active and interactive *Error-Tolerant Graph Matching* in which an active module queries one of the nodes of the graphs and the oracle (human or artificial) returns the node of the other graph it has to be imposed over the current correspondence between the involved graphs to improve it, the main aim of this model is to present an interactive framework that after a few interactions the model achieve the "perfect" correspondence between graphs.

In Chapter 4 it is described the problem of *Error-Tolerant Graph Matching* using substitution weights, and a method to automatically learn these weights such that the *hamming distance* (section 1.2.1.2) between a ground truth correspondence between graphs and the automatically obtained correspondence is minimised similar to [39]. Experimental results show that depending of the database we obtain a higher improvement than the compared method while learning the weights.

On the other hand, in Chapter 5, it is presented a method to learn the values for the insertion and deletion costs on nodes and edges without a human interaction such that the *hamming distance* between a ground truth node correspondence and the automatically obtained correspondence is minimised. This is because, in some applications, graphs in the reference and test databases are not split in classes and so it is not valid to minimise the classification accuracy. This is the case of some graph retrieval applications in which the aim is to find similar graphs without a previous classification. Moreover, there are some graph databases such that nodes and edges have only one or any attributes. In these cases, it has non-sense to learn the substitution weights for each feature as in [39, 40] but it is crucial to learn the best combinations of insertion and deletion costs on nodes and edges as unique Real numbers. Finally, we want the classical graph matching algorithms to be applied on our obtained costs. For this reason, the computed values have to be Real numbers and therefore, methods [37, 38] are not valid.

In Chapter 6, it is presented and evaluated an indexing scheme, modelled by a metric-tree, in which the cluster knowledge embedded in each node of the metric-tree is represented by one of the six *Graph-Class Prototypes* presented in the literature. The different representations of *Graph-Class Prototypes* are: *Set Median Graph* [54]; *Generalise Median Graph* [59, 58, 61, 62] synthesised through a *hierarchical method* [63], through a *genetic algorithm* [64] or through an extension of the *Graduated Assignment* algorithm [65]; *First-Order Random Graphs* [22]; *Function-Described Graphs* [23, 66]; *Second-Order Random Graphs* [24] and *Closure Graphs* [55]. Moreover, are evaluated two types of graph queries; the ones that the user imposes the graphs to be queried and the ones that the user imposes the maximum distance between the query graph and the returned graphs.

Finally, Chapter 7 explores the idea of bringing interactivity to the domain of 3D points alignment on a cooperative robotics framework. It is presented a new model in order to cooperatively improve the pose estimation in a fleet of robots.

Chapter 2
On the Relevance of Structural Dissimilarity Metrics on Graph Edit Distance

As we have seen in Chapter 1, graphs offer a convenient way to formally model objects or patterns.

Note that there are several kinds of objects that can be represented using graphs, in which the structural relations between the local components play a different role. For this reason, it is very interesting to analyse how it affects the fact of considering individually these local components and its structural relations to estimate the structural dissimilarities between them. At this point, a question arises. Which is the best structure? It seems that considering a large structure, we keep more structural information and the distance is less sub-optimal, nevertheless, the computational cost increases. In this chapter, four different local structures depending on the amount of structural information which is taken into account are defined below. Then, eight metrics to estimate the dissimilarity between the different local structures are presented.

Finally, to visualise the impact of the different metrics on the *Graph Edit Distance*, we show the classification accuracy results achieved by each metric in different databases using a nearest-neighbour classifier.

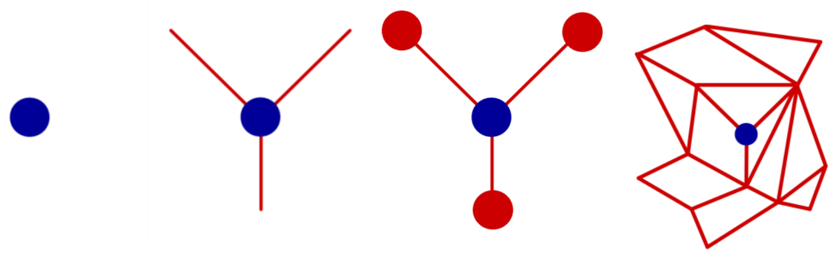
2.1 Moving from global to local structure to solve the Graph Edit Distance problem

In Chapter 1, section 1.2, we have seen that the local structure of a node in an *attributed graph* is termed N_{v_i}. In this chapter we will present different definitions for N_{v_i}.

The first definition of N_{v_i} is the empty set. That is, no structural information of node v_i is taken into account. Formally, $N_{v_i}^{Node} = \left(\Sigma_v^{N_{v_i}^{Node}}, \Sigma_e^{N_{v_i}^{Node}}, \gamma_v\right)$ with $\Sigma_v^{N_{v_i}^{node}} = \Sigma_e^{N_{v_i}^{node}} = \{\}$. We refer to this type of local structure as *Node*.

The second definition of N_{v_i} is given by $N_{v_i}^{Degree} = \left(\Sigma_v^{N_{v_i}^{Degree}}, \Sigma_e^{N_{v_i}^{Degree}}, \gamma_v\right)$ with $\Sigma_v^{N_{v_i}^{Degree}} = \{\}$ and $\Sigma_e^{N_{v_i}^{Degree}} = \{e_{ij} \in \Sigma_e\}$. In this case we regard the incident edges of v_i as local structure only (without adjacent nodes). This second definition of N_{v_i} is referred to as *Degree* from now on. The third definition of N_{v_i} used is given by the set of nodes that are directly connected to v_i including all edges that connect these nodes with v_i. Formally, $N_{v_i}^{Star} = \left(\Sigma_v^{N_{v_i}^{Star}}, \Sigma_e^{N_{v_i}^{Star}}, \gamma_v\right)$ with $\Sigma_v^{N_{v_i}^{Star}} = \{v_j | e_{ij} \in \Sigma_e\}$ and $\Sigma_e^{N_{v_a}^{Star}} = \{e_{ij} \in \Sigma_e | v_j \in \Sigma_v^{N_{v_a}^{Star}}\}$. We refer to this definition of local structure as *Star*. The last definition proposed is given by the whole edges of the graph (without nodes). We refer to this definition as *Mesh*. Formally, $N_{v_i}^{Mesh} = \left(\Sigma_v^{N_{v_i}^{Mesh}}, \Sigma_e^{N_{v_i}^{Mesh}}, \gamma_v\right)$ with $\Sigma_v^{N_{v_i}^{Mesh}} =, \{\}$ and $\Sigma_e^{N_{v_i}^{Mesh}} = \{\Sigma_e\}$. Note that the *Mesh* structure is an extension of the *Degree*. In Figure 2.1 an illustration of the four different local structures (*Node*, *Degree*, *Star* and *Mesh*) is shown.

Clearly, it is possible to define more levels of complexity. For instance, we could also have considered the whole edges and nodes connected to the adjacent nodes. But we decided not to consider them due to computational cost reasons, since the combinations of structures exponentially explode.



a. *Node* b. *Degree* c. *Star* d. *Mesh*

Figure 2.1. *Degree*, *Star* and *Mesh* structures (shown in red) of a single *Node* (shown in blue).

2.2 Distances between local structures

As has been commented on Chapter 1, section 1.3.1, a widely used method to evaluate the dissimilarity between two *attributed graphs* is the *Graph Edit Distance*. This dissimilarity model is to define a distance between two graphs G^p and G^q by means of the minimum amount of distortion required to transform G^p into G^q. Edit cost functions are typically introduced to quantitatively evaluate the level of distortion of each individual edit operation.

In this section, different methods inspired by the structures discussed in section 2.1 are proposed to determine costs $C_{i,a}$, $C_{i,\varepsilon}$ and $C_{\varepsilon,a}$, which are used in some of the most popular *Graph Edit Distance* algorithms [25, 67, 68] such as *Bipartite Graph Matching* reviewed in Chapter 1, section 1.3.2.1, of this thesis. These three costs depend on two weighted disjoint costs. The first one only depends on the nodes and the second one depends on the rest of the local structure.

We propose the following eight metrics to estimate the cost of structural dissimilarities.

2.2.1 Node

If the structural information of the nodes is not considered, that is we employ the *Node* local structure, we have

$$C(N_{v_i^p}^{Node} \rightarrow N_{v_a^q}^{Node}) = C(N_{v_i^p}^{Node} \rightarrow \varepsilon) = C(\varepsilon \rightarrow N_{v_a^q}^{Node}) = 0 \qquad \textbf{(2.1)}$$

2.2.2 Degree

For BP [25] as well as for SFBP [67] the same definition of the local structure of a node v has been used, viz. N_v is defined to be the set of incident edges of node v. That is, the *Degree* local structure, as formally described in section 2.1, is employed. Remember that in this thesis unlabeled edges are considered only, and thus, edge substitution is free of cost. Hence, using this definition of a local structure, the cost of substituting two local structures with each other is given by the difference of the numbers of edges of the involved nodes. Formally,

$$C(N_{v_i^p}^{Degree} \rightarrow N_{v_a^q}^{Degree}) = K_e \cdot \left|E(v_i^p) - E(v_a^q)\right| \qquad \textbf{(2.2)}$$

where K_e refers to a positive constant cost for deleting/inserting edges and *E(.)* refers to the number of edges of a certain node.

Likewise, the deletion and insertion costs of local structures depend on the number of incident edges $E(v_i^p)$ / $E(v_a^q)$ of the deleted or inserted node v_i^p and v_a^q, respectively. Formally,

$$C(N_{v_i^p}^{Degree} \to \varepsilon) = K_e \cdot E(v_i^p) \tag{2.3}$$

$$C(\varepsilon \to N_{v_a^q}^{Degree}) = K_e \cdot E(v_a^q) \tag{2.4}$$

We name this particular definition of the cost for processing local structures as *Degree*. We will use this metric for local structures as basic reference system.

Then, more definitions of the cost for processing local structures are presented. These definitions are based on the *Star* and *Mesh* local structures.

2.2.3 Star

The following four definitions of costs based on the *Star* structure assume that the complete local structure has to be inserted or deleted when the corresponding node is inserted or deleted, respectively. That is, insertion and deletion costs consider the cost of processing all edges that connect the central node and the cost of processing all adjacent nodes. Formally, the complete deletion and insertion costs of local structures depend on the number of adjacent nodes.

$$C(N_{v_i^p}^{Star} \to \varepsilon) = (K_v + K_e) \cdot E(v_i^p) \tag{2.5}$$

$$C(\varepsilon \to N_{v_a^q}^{Star}) = (K_v + K_e) \cdot E(v_a^q) \tag{2.6}$$

Where $(K_v + K_e)$ refers to the cost of deleting or inserting one edge and one node.

The substitution cost $C(N_{v_i^p}^{Star} \to N_{v_a^q}^{Star})$ for *Star* local structure is based on computing a distance between $N_{v_i^p}^{Star}$ and $N_{v_a^q}^{Star}$. For this particular computation every adjacent node and its corresponding edge is interpreted as an indivisible entity. That is, a *Star* local structure can be seen as a set of independent entities (nodes with adjacent edge), and thus, the computation of $C(N_{v_i^p}^{Star} \rightarrow N_{v_a^q}^{Star})$ refers to an assignment problem between two independent sets. This assignment problem can be solved in the same way as it is done with complete graphs by means of a cost matrix that considers substitutions, insertions and deletions. Formally,

$$C(N_{v_i^p}^{Star} \rightarrow N_{v_a^q}^{Star}) = AssignmentCost\left(N_{v_i^p}^{Star}, N_{v_a^q}^{Star}\right) \quad \textbf{(2.7)}$$

In order to compute this assignment cost we use four different assignment algorithms.

The first algorithm is given by an optimal algorithm for general *Linear Assignment Problems* known as *Hungarian* algorithm [68] that runs in cubic time. The second algorithm solves the assignment of $N_{v_i^p}^{Star}$ and $N_{v_a^q}^{Star}$ by means of the *Hausdorff* distance for subsets [28]. This assignment algorithm estimates the distance between two sets of entities by removing the restriction of finding a bijective mapping between the individual elements. In contrast with the *Hungarian* algorithm, *Hausdorff* assignments can be computed in quadratic time.

The third algorithm computes the dissimilarity between $N_{v_i^p}^{Star}$ and $N_{v_a^q}^{Star}$ by finding a suboptimal assignment of the individual entities by means of the *Greedy* assignment algorithm [68].

Finally, we propose to use a *Planar* distance metric. In this case, the relative position of each local structure is considered. That is, the only allowed assignments of entities from $N_{v_i^p}^{Star}$ to entities of $N_{v_a^q}^{Star}$ are the ones that are generated from cyclic combinations of the neighbour nodes. Formally, the sets $N_{v_i^p}^{Star}$ and $N_{v_a^q}^{Star}$ are interpreted as strings and the assignment cost is computed through the *Levenshtein* distance on these strings [70].

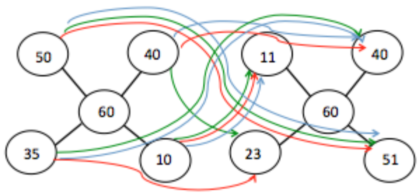


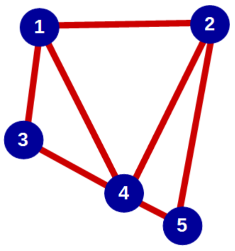
Figure 2.2. Optimal correspondences between two Star local structures using the different solvers proposed above to solve the assignment problem. *Hungarian* (red). *Planar* (green). *Hausdorff* (blue).

Figure 2.2 is an example of matching two *Star*s. Arrows show the optimal correspondences using *Hungarian* (red), *Planar* (green) and *Hausdorff* (blue) assignment solvers. Numbers on the nodes are attributes $\gamma_v^p(v_i^p)$ and $\gamma_v^q(v_a^q)$. Edges do not have attributes and $C(v_i^p \to v_a^q) = |\gamma_v^p(v_i^p) - \gamma_v^q(v_a^q)|$. In the *Hungarian* case (red arrows), $C\left(N_{v_i^p}^{Star} \to N_{v_a^q}^{Star}\right) = 1 + 0 + 1 + 12 = 14$. In the *Planar* case (green arrows), $C\left(N_{v_i^p}^{Star} \to N_{v_a^q}^{Star}\right) = 1 + 17 + 1 + 5 = 24$. The restriction to be the matching cyclic makes the distance value to be larger. And in the *Hausdorff* case (blue arrows), $C\left(N_{v_i^p}^{Star} \to N_{v_a^q}^{Star}\right) = 1 + 0 + 1 + 5 = 7$. Note the matching in the *Hausdorff* is not bijective. In the *Greedy* case, the matching is always bijective but it depends on the order of presentation of the nodes. If we suppose the order of the adjacent nodes in the first structure is {50, 40, 10, 35} then we obtain the same matching and distance than the *Hungarian* (red arrows).

2.2.4 Mesh

The last two definitions of costs use the *Mesh* structure. The basic idea is to compute the costs using the differences between scores achieved by two centrality models, the *Eigenvector* and the *Pagerank* [71], determining the importance of a node in a graph using the *Mesh* structure.

The first centrality is the *Eigenvector*. This centrality assigns a relative score to the nodes of the graph depending on the number of incident edges and the score of the neighbour nodes.



$$\rightarrow \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Figure 2.3. An adjacency matrix of a graph representing the *Mesh* of the graph (Square matrix used to represent pairs of nodes connected by an edge or not using binary code).

The eigenvector centralities given a specific node in both graphs are defined by,

$$\psi_i^p = \frac{1}{\lambda^p} \cdot \sum_{\{e_{ij} \in N_{v_i^p}^{Mesh} | v_j \in N_{v_i^p}^{Mesh}\}} |V_j^p| \quad \text{and} \quad \psi_a^q = \frac{1}{\lambda^q} \cdot \sum_{\{e_{ab} \in N_{v_a^q}^{Mesh} | v_b \in N_{v_a^q}^{Mesh}\}} |V_b^q| \tag{2.8}$$

where V_j^p and V_b^q are the values of the j-th and b-th positions of the eigenvectors with the largest eigenvalues obtained through adjacency matrices A^p and A^q (Figure 2.3). Besides, λ^p and λ^q are the largest eigenvalues of these adjacency matrices. Thus, the substitution cost is simply computed as,

$$C(N_{v_i^p}^{Mesh} \rightarrow N_{v_a^q}^{Mesh}) = |\psi_i^p - \psi_a^q| \tag{2.9}$$

The deletion and insertion costs are computed assuming that the centrality of a null node is 0.

$$C(N_{v_i^p}^{Mesh} \rightarrow \varepsilon) = \psi_i^p \tag{2.10}$$

$$C(\varepsilon \rightarrow N_{v_a^q}^{Mesh}) = \psi_a^q \tag{2.11}$$

The second centrality is the *Pagerank*. This centrality is a variation of the *Eigenvector* centrality. The difference consists in that the *Eigenvector* is normalised by the number of neighbours of the node.

$$\psi_i^p = \frac{1}{\lambda^p} \cdot \sum_{\{e_{ij} \in N_{v_i^p}^{Mesh} | v_j \in N_{v_i^p}^{Mesh}\}} \frac{\left|V_j^p\right|}{\max\left(\{1, E\left(v_j^p\right)\}\right)}$$

and **(2.12)**

$$\psi_a^q = \frac{1}{\lambda^q} \cdot \sum_{\{e_{ab} \in N_{v_a^q}^{Mesh} | v_b \in N_{v_a^q}^{Mesh}\}} \frac{\left|V_b^q\right|}{\max\left(\{1, E\left(V_b^q\right)\}\right)}$$

This centrality is normalised by $max\left(1, E\left(V_j^p\right)\right)$ instead of $E\left(V_j^p\right)$ to avoid dividing by 0. Then, the substitution, insertion and deletion costs are computed in a similar why than the eigenvector centrality but using the *PageRank*.

2.3 Experimental Evaluation

Database	**Ø nodes**	**Ø edges**	**Density**
LETTER-LOW	4.7	3.1	0.36
LETTER-MED	4.7	3.2	0.37
LETTER-HIGH	4.7	4.5	0.51
GREC	11.5	12.2	0.21
COIL-RAG	3	3	1
FINGERPRINT	5.42	4.42	0.26
PROTEINS	32.6	62.1	0.12

Table 2.1. Summary of graph data set characteristics (mean number of nodes, mean number of edges, and mean graph density).

In total, we have eight different cost models to compute the individual entries $c_{i,a}$, $c_{i,\varepsilon}$ and $c_{\varepsilon,a}$. The aim of the first experiment is to show the quality of the obtained correspondences given the SFBP algorithm to compute the GED with the eight cost models described above and to evaluate this quality together with the runtime. Figure 2.4 and Figure 2.5 show the mean cost and runtime given 10 rounds. In each round, we randomly generated 200 pairs of graphs of order 10 with a specific density, one attribute on the nodes (natural number from 0 to 99) and unattributed edges. The correspondences has been obtained given the eight cost models with $K_v = K_e = 50$. Finally, given this correspondences, we computed the cost using the equation (1.2).

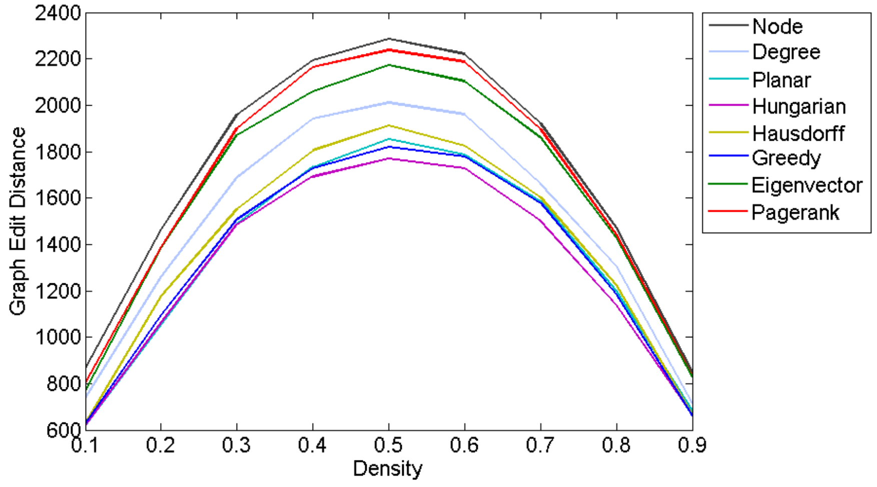


Figure 2.4. Mean sub-optimal GED for the SFBP and the eight cost models, increasing the density of the randomly generated graphs. Density=0: no edges. Density=1: completely connected.

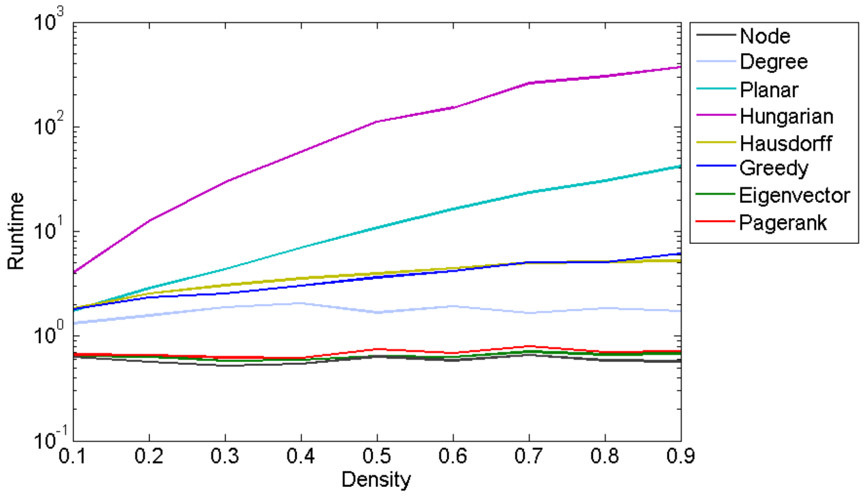


Figure 2.5. Mean runtime logarithmically scaled in seconds to compute the SFBP in the experiments shown in Figure 2.4.

When the density is near to 0.5, more edges are deleted or inserted and for this reason, the whole correspondences tend to obtain larger GEDs.

Moreover, the presence of edges in the nodes is relevant to find structural similarities and so the differences between the obtained GEDs increase. If we focus on the runtime; *Eigenvector*, *Pagerank*, *Nodes* and *Degree* are constant with respect to the graph density (the figure is represented in logarithmic scale). In general, the centralities that obtain lower GED (the best ones) are the slowest ones.

The aim of the second experiment is to compare SFBP using the eight cost models. Table 2.2 and Table 2.3 show the recognition ratio and mean runtime. We tested seven graph databases *LETTER-LOW*, *LETTER-MEDIUM*, *LETTER-HIGH*, *GREC*, *COIL-RAG*, *FINGERPRINT* and *PROTEINS* from the IAM repository [72] considering unlabelled edges, these databases are described in detail in the Databases annex of this thesis. We used the KNN classifier where $K = 3$. The *Jonker–Volgenant* solver [31] has been used on the *graph matching* algorithm SFBP and the cyclic string matching [70] to compute the *Planar* assignment for *Star* local structure. Note in SFBP, the computation of the costs $C_{i,a}$, $C_{i,\varepsilon}$ and $C_{\varepsilon,a}$ depends on the following parameters: In case of *Degree* and *Star* local structures, parameters are K_v, K_e and parameter β is fixed to 0.5 for this local structures. In case of *Mesh*, parameters are K_v and β. We show the recognition ratio given four different costs configurations because we want to study the relevance of these costs in terms of accuracy and runtime. These values are the usual ones used to perform classification tasks in each dataset [167]. Given a database, best accuracies are marked in bold. And given a cost model and a database, best accuracies are underlined. Observing the achieved results in Table 2.2 and Table 2.3 we note that the different cost models configurations for the SFBP present a different performance depending on the database. We assume that it is because graphs that integrate each database have different topologies.

In Table 2.1 we summarise the mean densities and the mean number of nodes and edges for each database. We want to relate these characteristics to the performance of the different centralities in terms of accuracy and runtime presented in Table 2.2 and Table 2.3. When the density is very low (*PROTEINS*) or very high (*COIL-RAG*) there is poor structural information and then centralities tend to obtain similar accuracies. This is due to the edit distance tends to be similar as shown in Figure 2.4. In this case, we recommend using the *Node* cost model because it is the fastest one. In sparse graphs (low densities) (*GREC*), the local structural information little contributes on the global structure and for this reason, the local structures that consider the whole model obtain the highest accuracies, such as the *Eigenvector* and *Pagerank*. But note β parameter influences the runtime on these costs models.

	Parameters		***Node***	***Degree***	***Star***				***Parameters***		***Mesh***	
	K_v	K_e	-	-	***Planar***	***Hungarian***	***Hausdorff***	***Greedy***	K_v	β	***Eigenvector***	***Pagerank***
LETTER LOW	**1**	**1**	0.90	0.95	0.98	0.98	0.97	0.98	**1**	**0.7**	0.95	0.95
	1	**0.1**	0.90	0.96	0.98	**0.99**	**0.99**	0.98	**1**	**0.5**	0.96	0.96
	0.1	**1**	0.80	0.96	0.98	0.98	0.98	0.98	**1**	**0.3**	0.95	0.95
	0.1	**0.1**	0.80	0.98	**0.99**	**0.99**	**0.99**	**0.99**	**1**	**0.1**	0.92	0.92
LETTER MED	**1**	**1**	0.59	0.87	**0.94**	**0.94**	0.93	0.92	**1**	**0.7**	0.79	0.77
	1	**0.1**	0.59	0.73	**0.94**	**0.94**	**0.94**	0.92	**1**	**0.5**	0.85	0.84
	0.1	**1**	0.08	0.86	0.89	0.89	0.89	0.88	**1**	**0.3**	0.88	0.87
	0.1	**0.1**	0.08	0.18	0.09	0.09	0.09	0.10	**1**	**0.1**	0.83	0.83
LETTER HIGH	**1**	**1**	0.65	0.79	**0.90**	0.88	**0.90**	0.79	**1**	**0.7**	0.69	0.68
	1	**0.1**	0.65	0.74	**0.90**	0.88	0.89	0.82	**1**	**0.5**	0.72	0.71
	0.1	**1**	0.08	0.81	0.80	0.80	0.80	0.80	**1**	**0.3**	0.76	0.76
	0.1	**0.1**	0.08	0.38	0.07	0.07	0.07	0.15	**1**	**0.1**	0.59	0.61
GREC	**10**	**10**	0.58	0.96	0.83	0.84	0.85	0.91	**10**	**0.7**	0.87	0.87
	10	**5**	0.58	0.92	0.63	0.65	0.66	0.89	**10**	**0.5**	0.93	0.93
	5	**10**	0.18	0.95	0.37	0.39	0.40	0.76	**10**	**0.3**	0.97	0.97
	5	**5**	0.18	0.87	0.14	0.14	0.14	0.42	**10**	**0.1**	0.97	0.96
COIL RAG	**1**	**1**	**0.93**	0.89	0.90	0.89	0.89	0.90	**1**	**0.7**	0.92	0.92
	1	**0.1**	**0.93**	0.92	0.90	0.90	0.91	0.91	**1**	**0.5**	0.92	0.91
	0.1	**1**	0.87	0.89	0.90	0.90	0.90	0.91	**1**	**0.3**	0.90	0.90
	0.1	**0.1**	0.87	**0.93**	0.91	0.91	0.91	0.91	**1**	**0.1**	0.88	0.88
FINGERPRINT	**50**	**50**	**0.65**	**0.65**	0.64	0.64	0.64	**0.65**	**50**	**0.7**	0.64	0.64
	50	**5**	**0.65**	**0.65**	0.64	0.64	**0.65**	**0.65**	**50**	**0.5**	0.64	0.64
	5	**50**	0.29	**0.65**	0.63	0.63	0.63	0.64	**50**	**0.3**	0.64	0.64
	5	**5**	0.29	0.49	0.37	0.37	0.37	0.40	**50**	**0.1**	0.63	0.63
PROTEINS	**1**	**1**	0.46	0.46	0.44	0.46	0.47	0.50	**1**	**0.7**	0.46	0.44
	1	**0.1**	0.46	0.49	0.41	0.43	0.45	**0.52**	**1**	**0.5**	0.46	0.44
	0.1	**1**	0.42	0.45	0.37	0.38	0.43	0.49	**1**	**0.3**	0.44	0.42
	0.1	**0.1**	0.42	0.43	0.32	0.31	0.33	0.38	**1**	**0.1**	0.42	0.44

Table 2.2. Recognition ratio of the proposed cost models to estimate the structural dissimilarities using SFBP given the 7 datasets and different combinations of parameters.

In the medium densities (*LETTER*), the local structure becomes the most relevant. This is due to the gap between distances maximises at medium densities (Figure 2.4). Then, the cost models based on the *Star* local structure achieve the best accuracies. We can decide which one to be used in this case depending on the compromise between accuracy and runtime. The *Hungarian* is the slowest one (Figure 2.5 and Table 2.3) but achieves the best accuracy in some cases (Table 2.2). Finally, although the *FINGERPRINT* database has a medium density, the structural information of this database with unlabelled edges does not contribute enough to improve the accuracy with respect to the *Node*. For this reason, accuracies in this database are almost equal.

	Parameters		*Node*	*Degree*	*Star*				*Parameters*		*Mesh*	
	K_v	K_e	-	-	*Planar*	*Hungarian*	*Hausdorff*	*Greedy*	K_v	β	*Eigenvector*	*Pagerank*
LETTER LOW	**1**	**1**	0.67	0.61	0.95	3.14	1.60	1.33	**1**	**0.7**	0.95	0.95
	1	**0.1**	0.61	0.56	0.90	3.15	1.40	1.23	**1**	**0.5**	0.96	0.96
	0.1	**1**	0.48	0.55	0.85	3.09	1.26	1.08	**1**	**0.3**	0.95	0.95
	0.1	**0.1**	**0.46**	0.48	0.80	3.15	1.18	1.01	**1**	**0.1**	0.92	0.92
LETTER MED	**1**	**1**	0.63	0.61	0.94	3.13	1.44	1.36	**1**	**0.7**	0.79	0.77
	1	**0.1**	0.71	0.56	0.91	3.16	1.74	1.24	**1**	**0.5**	0.85	0.84
	0.1	**1**	0.48	0.55	0.86	3.09	1.25	1.09	**1**	**0.3**	0.88	0.87
	0.1	**0.1**	**0.47**	0.49	0.81	3.16	1.19	0.98	**1**	**0.1**	0.83	0.83
LETTER HIGH	**1**	**1**	0.74	0.61	1.24	3.68	1.76	1.74	**1**	**0.7**	0.69	0.68
	1	**0.1**	0.70	0.60	1.26	3.60	1.60	1.68	**1**	**0.5**	0.72	0.71
	0.1	**1**	**0.46**	0.57	1.17	3.52	1.35	1.20	**1**	**0.3**	0.76	0.76
	0.1	**0.1**	0.47	0.47	1.10	3.51	1.24	1.10	**1**	**0.1**	0.59	0.61
GREC	**10**	**10**	0.38	0.45	1.99	7.34	2.12	1.19	**10**	**0.7**	0.87	0.87
	10	**5**	0.40	0.42	2.01	7.48	2.08	1.80	**10**	**0.5**	0.93	0.93
	5	**10**	0.36	0.42	1.99	7.35	1.92	1.61	**10**	**0.3**	0.97	0.97
	5	**5**	**0.34**	0.40	1.97	7.34	1.78	1.48	**10**	**0.1**	0.97	0.96
COIL RAG	**1**	**1**	2.75	4.77	6.31	15.61	5.94	5.45	**1**	**0.7**	0.92	0.92
	1	**0.1**	2.76	3.60	5.66	13.17	4.92	4.77	**1**	**0.5**	0.92	0.91
	0.1	**1**	1.86	3.77	4.21	11.10	3.91	3.91	**1**	**0.3**	0.90	0.90
	0.1	**0.1**	**1.22**	2.84	3.32	6.45	2.22	2.10	**1**	**0.1**	0.88	0.88
FINGERPRINT	**50**	**50**	0.89	3.36	2.54	6.36	2.47	2.35	**50**	**0.7**	0.64	0.64
	50	**5**	0.83	1.67	1.88	5.44	1.89	2.30	**50**	**0.5**	0.64	0.64
	5	**50**	**0.28**	2.00	1.12	3.86	1.12	1.04	**50**	**0.3**	0.64	0.64
	5	**5**	**0.28**	0.39	0.95	3.72	0.88	0.74	**50**	**0.1**	0.63	0.63
PROTEINS	**1**	**1**	2.44	5.43	39.76	81.34	13.41	13.46	**1**	**0.7**	0.46	0.44
	1	**0.1**	2.83	7.03	54.13	97.73	14.87	14.98	**1**	**0.5**	0.46	0.44
	0.1	**1**	2.24	7.33	60.23	98.66	17.15	16.84	**1**	**0.3**	0.44	0.42
	0.1	**0.1**	**2.23**	4.26	58.42	81.05	15.29	15.65	**1**	**0.1**	0.42	0.44

Table 2.3. Mean runtime spent for graph classification of the proposed cost models to estimate the structural dissimilarities using SFBP given the 7 datasets and different combinations of parameters (Matlab, i7 950, 3.07 GHz, 6 GB RAM, Windows 7).

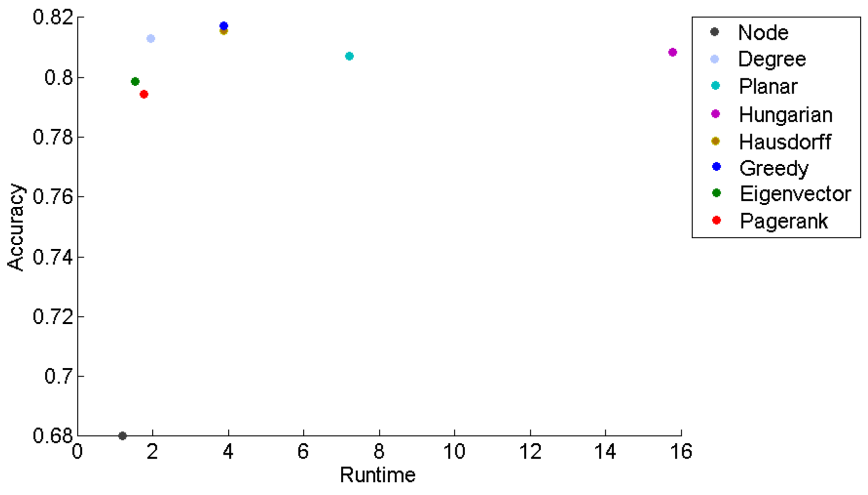


Figure 2.6. Accuracy with respect to runtime in seconds.

Figure 2.6 shows the mean of the best accuracies with respect to the mean runtime of the eight methods presented in Table 2.2 and Table 2.3 throughout the seven databases. Given a method and a database Table 2.2 and Table 2.3 show four results. From these four results we only have taken the one that obtained the maximum accuracy and its corresponding runtime, since it would be the one selected by an expert in a validation process. These last results have been show to analyse the general behaviour of the cost models independently of the topology of the databases. In general, we realise *Node* obtains a so low accuracy that does not compensate on the low runtime. Contrarily, *Stars* using *Planar* and *Hungarian* solvers have an interesting accuracy but they are too slow. Moreover, centralities based on *Mesh* are fast but do not achieve the best accuracies. In fact, several previous experimental validations on pattern recognition have shown us that these methods are too dependent on intraclass variations. All in all, from the results represented in this figure we could conclude the best centralities to be considered are the *Degree*, and *Star* using *Hausdorff* and *Greedy* assignment solvers.

Chapter 3
Interactive Graph Matching using Active Query Strategies

In Chapter 1 we introduced the *Error-Tolerant Graph Matching* [5] problem (section 1.3). Due to distortions of the data and the complexity of the problem, in some applications, completely automatic processes do not return a satisfactory correspondence. In this chapter it is proposed a framework in which an active module, queries one of the nodes of one graph and an oracle (human or artificial) to impose a node of the other graph to be mapped. For each oracle node-to-node imposition, other mappings are automatically amended. The framework can be implemented over any graph matching algorithm that iteratively updates a probability matrix (see section 1.3.2.2) between nodes since it only requires access to the probability matrix and to update the costs between nodes and edges of the graph matching method (*Graduated Assignment* [31] in our case).

3.1 Overview of interactive Graph Matching

On one hand, *active learning* is a discipline concerned with the design and development of algorithms that allow computers to evolve behaviours based on examples [73, 74]. In this discipline, a learner can take advantage of examples to capture characteristics of interest from the data with respect to their class. With the learned characteristics, the learner deduces the class of the new examples. On the other hand, *Error-Tolerant Graph Matching* [5] is another discipline that aims to find the best correspondence between the nodes of both graphs so that the cost of this optimal correspondence is the minimum among all possible correspondence. If we combine the *Active Learning* and *Error-Tolerant Graph Matching* disciplines, we can define a model in which examples and classes in the machine-learning discipline are composed of the set of nodes of one of the graphs and the nodes of the other graphs, respectively. Therefore, what we want to find is the best correspondence between the nodes of both graphs but with the minimum necessary help of an oracle. Note that in this chapter we do not perform a learning strategy such as *Learning Graph Matching* [39] or *Semi-Supervised Learning* [127] since we do not modify the *matching cost* function.

Normally, two basic modules compose pattern recognition systems [73]. The first one extracts the main features given the raw data. The second one extracts the class of the object or simply obtains the most similar object from a database. In the semi-automatic methods, a specialist usually interacts in the first module and modifies the automatically extracted features. Then, with the updated features, the automatic correspondence or query process is performed obtaining a result with higher quality. For instance, in the case of AFIS [20], the specialist usually verifies and modifies the extracted minutiae of the fingerprint to be queried. However, it is not so usual to apply any interaction on the second module [75, 76].

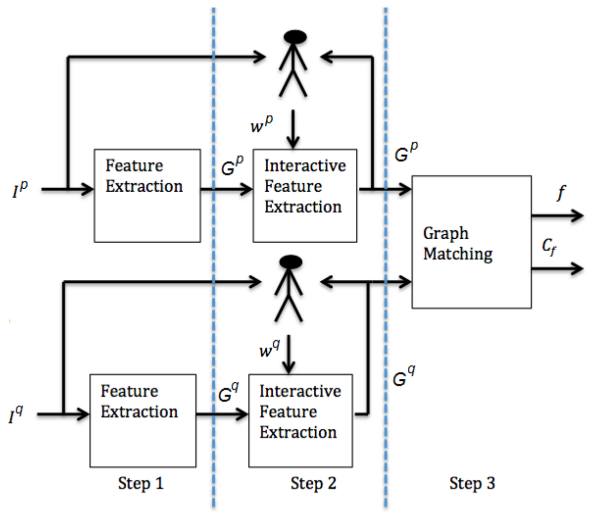


Figure 3.1. Semi-automatic image-correspondence process with human interaction in the local parts extraction and based on structural pattern recognition.

Figure 3.1 shows a scheme of semi-automatic image-correspondence process in which an intermediate step has been incorporated. We wish to compare input images I^p and I^q. Both images are represented by some kind of representation that explore the local parts of the image $G^p = G(I^p)$ and $G^q = G(I^q)$, for instance vectors or *attributed graphs*. There is a first step in which representations G^p and G^q of the images I^p and I^q are obtained using methods such as [77, 78]. Then, in the semi-automatic methods, there is a second step in which the oracle edits the local parts of these representations (erase, create or modify their positions or values). We call the oracle feedback w^1 and w^2. Note that the oracle not only has access to the obtained representation but also to the original image since it is a valuable knowledge. The last step obtains the correspondence between nodes f and a dissimilarity measure or cost C_f in a completely automatic way through methods such as [32, 33, 79, 80, 81, 82].

The aim of this chapter is to present a new model for add interactivity to the third step of Figure 3.1 and to keep the first and second steps as they are. This new interactive method is useful in two types of applications. The first ones are applications where it is crucial to have a perfect match but data is very noisy and it is difficult to extract the local parts of objects even though the number of these local parts may not be large. For instance, in medical applications, in which graphs are extracted from images. Other applications are the ones in which graphs have a large cardinality. Then, graph matching algorithms have to be very greedy and they are unlikely to obtain a satisfactory match. Conversely, this method is not useful in applications where an unclassified graph has to be compared with a large number of graphs in a database. For instance, fingerprint identification. The human interaction in each graph comparison would increase the run time considerably. In this type of application, it is usual to interact in the first step of the pattern recognition process as described in Figure 3.1. The extracted graph is corrected in the first step of the recognition process and graphs of the database are corrected in the enrolment process.

The framework it is designed to be implemented over any graph matching algorithm that computes a probability matrix (see section 1.3.2.2). In section 3.3, it is shown how the feedback of an oracle can be used to update matrices C_v and C_e. Besides, we present different strategies which, with the information of the probability matrix P and the magnitude of the gradient M, derive the node that has to be queried to the oracle.

The rest of the chapter is organised as follows. In 3.2, it is presented the active and interactive learning models; in other words, it is shown how to add interactivity to the third step of the image correspondence process (Figure 3.1). In 3.3, it is shown the algorithm to compute the active and interactive graph matching. Finally, in 3.4 it is shown the practical evaluation.

3.2 Active Learning

The key idea behind *active learning* [84, 85, 86] is that a machine learning algorithm can achieve a greater accuracy with fewer classified training examples if it is allowed to choose the data from which it learns. The learner queries some elements and the answerer of the queries decides which classes these elements belong to. The answerer might be another automatic system or a human annotator and in general, it is called an oracle since it is assumed its answer is always correct.

Active learning is well motivated in many modern machine-learning problems, where unclassified examples may be abundant but finding the class is difficult, time-consuming or expensive to obtain [73, 76]. *Active learning* has been applied in several fields such as speech recognition [87], information extraction [88, 89, 90, 91], robotics [92], transcription of text images [93, 94], object classification in general [95, 96, 97, 98, 99], biometrics [124], image segmentation [125], clustering [126]. And in general, it has been used for parameter selection [100].

3.2.1 Active Learning scenarios

3.2.1.1 Membership query synthesis

In this scenario, the learner may request the class for any unclassified instance in the input space, including and assuming instances where the learner synthetically generates [99, 101, 102, 103]. Query synthesis is reasonable for many problems but classifying such arbitrary instances can be awkward if the oracle is a human annotator. For instance, in [104], they employed this method to train a neural network to classify handwritten characters and they encountered the problem that most of the generated instances were not recognisable symbols for a human annotator.

3.2.1.2 Stream-based Selective Sampling

In this scenario, each unclassified instance is typically drawn one at a time from the input space and the learner must decide whether to query or discard it depending on some informativeness measure (also called query strategy) [106]. One approach is to compute an explicit region of uncertainty [106, 107], i.e., the part of the input space that it is still ambiguous to the learner and the learner only queries instances that fall within it. Another approach is to make a biased random decision, so that more informative instances are, more likely to be queried [108].

3.2.1.3 Pool-based Active Learning

In this scenario, large collections of unclassified data can be gathered at once [95, 96, 109, 110, 111]. It assumes there is a small set of classified data and a large pool of unclassified data available. Thus, instances from the unclassified pool are queried in a greedy fashion, according to an informativeness measure used to evaluate all instances in the unclassified pool. The main difference between stream-based and pool-based active learning is that the former receives the instances sequentially and makes query decisions individually, whereas the latter evaluates and ranks the entire collection of unclassified instances and selects the best instance to be queried.

3.2.2 Query Strategies

Moreover, all active learning scenarios involve evaluating the informativeness of unlabelled instances. There have been many proposed ways of formulating such query strategies [84] and this subsection provides an overview of the strategies used to date. We use the following notation. x is an unclassified instance that can be queried, y is one of the classes and θ is a classification model. Finally, the element x_A^* refers to the most informative instance according to some query strategy A. Besides, the conditional probability $P(y|x;\theta)$ represents the posterior class probability of class y given an instance x and a classification model θ.

3.2.2.1 Uncertainly Sampling

The active learner queries the instances about which it is least certain how to classify [110]. This approach is often straightforward for probabilistic learning models. For instance, when there are only two classes, the sampling strategy simply queries the instance whose posterior probability of a class is nearest ½. For the multiple class case, there are three interesting options.

3.2.2.1.1 <u>Least Confident</u>

This strategy [111] queries the element whose highest probability of belonging to a class is the lowest among all the elements.

$$x_{LC}^* = \underset{\forall x}{\operatorname{argmin}} P(y^*|x;\theta) \qquad \textbf{(3.1)}$$

where $y^* = \text{argmax}_{\forall y}\, P(y|x;\theta)$ is the most likely class labelling.

3.2.2.1.2 <u>Margin sampling</u>

This strategy [112] aims to incorporate the posterior probability of the second most likely labelled. Intuitively, instances with large margins are easy, since the classifier has little doubt in differentiating between the two most likely class labels. On the contrary, instances with small margins are more ambiguous, thus knowing the true label would help the model for discriminate more effectively between them. If y^1 and y^2 are the first and second most probable class labels under the model θ, respectively, the queried element is,

$$x^*_{MS} = \underset{\forall x}{\text{argmin}}\{P(y^1|x;\theta) - P(y^2|x;\theta)\} \tag{3.2}$$

3.2.2.1.3 <u>Maximum Entropy</u>

This strategy [113] queries the element with maximum *Shannon Entropy* [114] given the probabilities. The main idea of the method is to query the elements that are more difficult to be classified,

$$x^*_{ME} = \underset{\forall x}{\text{argmax}}\left\{-\sum_{\forall y} P(y|x;\theta)log\big(P(y|x;\theta)\big)\right\} \tag{3.3}$$

3.2.2.2 Query by committee

This strategy [115, 116, 117] involves maintaining a committee of models which are all trained on the current labelled set but represent competing hypothesis. The most informative query is considered to be the instance about which they disagree most. To measure a level of disagreement, one option is the vote entropy [108]

$$x^*_{QBC} = \underset{\forall x}{\text{argmax}}\left\{-\sum_{\forall y}\frac{V(y)}{C}log\left(\frac{V(y)}{C}\right)\right\} \tag{3.4}$$

where $V(y)$ represents the number of votes that a class receives from among the committee member's predictions and C is the number of committees.

3.2.2.3 Expected Model Change

An active learner queries the instances that would impart the greatest change to the current model if we knew its class [118]. Since probabilistic models are usually trained using a *Gradient Ascent* technique [34], the change imparted to the model can be measured by the magnitude of the training gradient. Given that the learning module does not know the true class y of an instance x in advance, we must instead calculate the length as an expectation over the possible classes. Moreover, we assume the resulting magnitude of the training gradient M when the pair $\langle x, y\rangle$ has been added to the model θ is similar to the gradient magnitude of the probability related to $\langle x, y\rangle$. We make this approximation because the gradient magnitude should be nearly zero given the method converged in the previous round training and because we assume the training instances are independent.

$$x^*_{EMC} = \underset{\forall x}{\operatorname{argmax}} \left\{ \sum_{\forall y} P(y|x;\theta) M(\langle x, y\rangle, \theta) \right\} \tag{3.5}$$

3.2.2.4 Variance Reduction

The aim of this strategy [103, 119], is to query the instance that minimises the learner's future error by minimising its variance. They used the estimated distribution of the model's output to estimate the variance σ^2_θ of the learner after some new instance x has been labelled to class y.

$$x^*_{VR} = \underset{\forall x}{\operatorname{argmin}} \{\sigma^2_\theta\} \tag{3.6}$$

3.2.2.5 Estimated Error Reduction

Similarly to *Variance Reduction* strategy, the aim of this strategy [120, 121, 122] is to query the instance that minimises the learner's future error but, instead of minimising the variance, the aim is to minimise the expectation of this error E_θ.

$$x_{VR}^{*} = \underset{\forall x}{\operatorname{argmin}}\{E_\theta\} \tag{3.7}$$

3.2.2.6 Density-Weighted methods

It has been suggested that *Uncertainty Sampling* and *Query by Committee* strategies are prone to querying outliers. Although they are the least certain instances, they are not representative of the other instances in the distribution. Therefore, knowing their label is unlikely to improve the accuracy of the data as a whole. These methods [111] are based on adding a weighting term which depends on the probability of the element being an outlier.

3.3 Interactive graph matching using active queries

In this section, it is explained how to add interactivity to the graph matching module. We define the input and output architecture and how we use the feedback to modify the nodes and edges costs. Secondly, are proposed different active strategies (inspired by the ones presented in 3.2.2) to decide which node has to be presented to the oracle.

3.3.1 Adding interactivity into the graph matching module

Human specialists are very good at finding the correspondences between local parts of an object, for instance, the minutiae of two fingerprints, but this is one of the most difficult tasks for an automatic system. When features that represent the object are structured on *attributed graphs*, we have an opportunity to have a second interaction on the system (at third step, Figure 3.1). The new model shifts from the concept of fully automatic graph matching to a model where obtained mapping is conditioned by the feedback. This shift is caused by the fact that the correspondence obtained by the fully automatic system often turns out to be non-natural. Moreover, since our algorithm modifies the automatically-obtained costs matrices between nodes C_v and edges C_e to be adapted to the feedback, then mistakes made at the first step of the structural pattern recognition are partially amended.

In the *Active and Interactive* model presented in this thesis, the oracle can recursively interact in the graph matching process until he considers a satisfactory correspondence has been reached. In each interaction, the automatic process uses the hypothesis imposed by the oracle and, considering the graph-distance model, obtains a new correspondence between nodes.

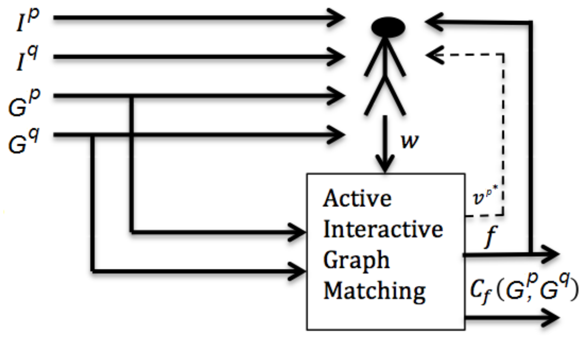


Figure 3.2. Interactive and Active Graph Matching Process (new step 3 of Figure 3.1).

Figure 3.2 shows a schematic view of the active and interactive graph matching process that substitutes step 3 of Figure 3.1. The oracle has access to the original images because they have more information. Moreover, the oracle also has access to both *attributed graphs* and the current correspondence f between graph nodes. The output of the module is the same as the classical graph matching: the obtained correspondence $f(G^p, G^q)$ and the cost related to this correspondence $C_f(G^p, G^q)$. The active module presents to the specialist the node v^{p^*} which is supposed to produce a greater impact on the correspondence between both graphs if its mappings was certainly known (dashed line in Figure 3.2). Thus, these queries help the specialist (who acts as the oracle in the machine-learning scenario) to decide which node mapping he wants to interact with. In the system we propose, the feedback w is not only the answer of the query v^{p^*} but also other suggestions of the current correspondence. This is because there is no sense in allowing the human to amend a node mapping if he has detected that it is wrong even though the active module does not detected that it might be wrong.

We represent the interactions as a vector $w = \left[w^{(1)}, \dots, w^{(k)}\right]$ where each position represents a simple user action $w^{(m)}$. In each interactive step of the algorithm, the user can interact with k different number of possible simple actions. When the user wants to impose a mapping $f\left(v_i^p\right) = v_a^q$, then the simple user action is $w^{(m)} = Set\left(v_i^p, v_a^q\right)$. Conversely, when the user accepts the current mappings for all the nodes, $v_i^p \to f\left(v_i^p\right) \forall i = \{1, \dots, n\}$, the interaction is $w^{(m)} = OK$.

Figure 3.3 shows an example of a current correspondence in green (obtained through the algorithm presented in [80]), the imposed actions in purple and the selected node in red.

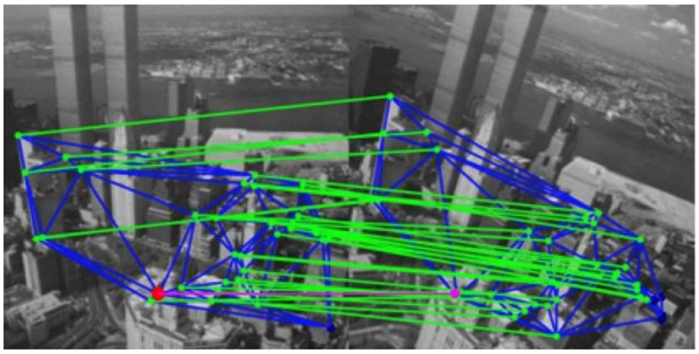


Figure 3.3. Graphical representation of the current correspondence in green and the feedback in purple.

If the feedback is $Set\left(v_i^p, v_a^q\right)$, then the node costs are modified as follows,

$$
\begin{aligned}
&C_v\left[v_i^p, v_a^q\right] = 0\\
&C_v\left[v_i^p, v_b^q\right] = \infty \forall b \neq a\\
&C_v\left[v_j^p, v_a^q\right] = \infty \forall j \neq i
\end{aligned}
$$

The aim is to force the graph matching algorithm to impose this mapping at the following iterations and also to impose the global matching to be a bijection.

In the case of edge costs, the modus operandi is similar. Nevertheless, we need the correspondence of both terminal nodes to be accepted by the user in this iteration or previous ones. If it is not the case, this action has no influence on the edge costs. Therefore, if $Set(v_i^p, v_a^q)$ and $Set\left(v_j^p, v_b^q\right)$ then

$$C_e\left[v_i^p, v_a^q, v_j^p, v_b^q\right] = 0 \wedge C_e\left[v_j^p, v_b^q, v_i^p, v_a^q\right] = 0$$
$$C_e\left[v_i^p, v_{a'}^q, v_j^p, v_{b'}^q\right] = \infty \wedge C_e\left[v_j^p, v_{b'}^q, v_i^p, v_{a'}^q\right] = \infty \ \forall a \neq a' \text{and} b \neq b'$$
$$C_e\left[v_{i'}^p, v_a^q, v_{j'}^p, v_b^q\right] = \infty \wedge C_e\left[v_{j'}^p, v_b^q, v_{i'}^p, v_a^q\right] = \infty \forall i \neq i' \text{and} j \neq j'$$

3.3.2 Active Learning strategies based on the probability matrix

In this section, we present several strategies to select a node v^{p^*} of G^p that have to be queried to the oracle. The oracle feedback is v^{q^*} which means he believes $f\left(v^{p^*}\right) = v^{q^*}$. From the three scenarios presented in section 3.2.1, the pool-based active learning is the one that can be addressed directly to our problem since we have access to all the elements to be classified (graph nodes of G^p) and also the predefined classes (graph nodes of G^q). Normally, query elements are selectively drawn from the set of unclassified elements. In our case, an “unclassified element” is a node of G^p whose mapping we do not know. For this reason, the pool of nodes to be queried is composed of nodes of G^p that have never been queried before. In the strategies we present, there is a logical function $Q(v_i^p)$that shows if node v_i^p has been queried before. This logical function is used to ensure a node is not queried several times. Note that in the case where $Q(i) = True$ for all nodes of G^p then the following strategies return an empty value.

We leave for future work to address the other two depicted scenarios. On the one hand, the membership query synthesis scenario would involve generating new nodes of G^p. It may be seen as the insert operation in the graph matching method based on edit costs [18]. On the other hand, the stream-based selective sampling scenario would be logical if we do not have access to the whole graph at once. For instance, in cases where graphs are dynamic and very large, such as social networks.

3.3.2.1 Uncertainly Sampling

We define four different strategies based on *Uncertainly Sampling*: *Least Confident*, *Least Confident given the Current Labelling*, *Margin Sampling* and *Maximum Entropy*.

3.3.2.1.1 Least Confident (LC)

The learner queries the node $v_{LC}^{p^*}$ of G^p that has not been previously queried and whose maximum probability given the nodes of G^q is the lower. Node $v_{LC}^{p^*}$ is obtained in two steps. Firstly, we obtain the set of nodes in G^q: $\{v^{q\{1\}}, \dots, v^{q\{i\}}, \dots, v^{q\{n\}}\}$ so that,

$$v^{q\{i\}} = \underset{\forall j=\{1,..,n\}}{\operatorname{argmax}} P\left[v_i^p, v_j^q\right];\ \forall i = \{1,..,n\} \tag{3.8}$$

note that $v^{q\{i\}}$ represents the node selected of G^q when v_i^p is considered. For this reason, some of the nodes in this set can appear several times, $v^{q\{i\}} = v^{q\{j\}}; i \neq j$.

And secondly, we select the node in G^p so that its respective node in the set obtains the minimum probability,

$$v_{LC}^{p^*} = \underset{\forall i=\{1,..,n\} | Q(i) = False}{\operatorname{argmin}} P\left[v_i^p, v^{q\{i\}}\right] \tag{3.9}$$

Computational cost: $O(n^2)$

3.3.2.1.2 Least Confident given the Current Labelling (LCCL)

The aim of this strategy is to query the nodes that are matched through the current correspondence but they have not been queried before. Therefore, it can be seen that the method tries to minimise the *hamming distance* between the current correspondence and the ideal correspondence (the correspondence that would have been predicted by the oracle if all the nodes were queried). The learner queries node $v_{LCCL}^{p^*}$ of G^p that has not been previously queried and has the minimum probability given the current correspondence f. Formally,

$$v_{LCCL}^{p^*} = \underset{\forall i=\{1,..,n\} | Q(i)=False}{\operatorname{argmin}} P\left[v_i^p, f\left(v_i^p\right)\right] \tag{3.10}$$

Computational cost: $O(n)$

3.3.2.1.3 Margin Sampling (MS)

Defining $v^{q\{i\}}$ as the most probable node with respect to v_i^p and it is defined as in equation (3.11). We also define $v^{q'\{i\}}$ as the second most probable node and defined in a similar way to $v^{q\{i\}}$ but without considering node $v^{q\{i\}}$. Thus, the queried element is,

$$v_{MS}^{p^*} = \underset{\forall i=\{1,..,n\} | Q(i)=False}{\operatorname{argmin}} \left\{P\left[v_i^p, v^{q\{i\}}\right] - P\left[v_i^p, v^{q'\{i\}}\right]\right\} \tag{3.11}$$

Computational cost: $O(n^2)$

3.3.2.1.4 Maximum Entropy (ME)

The selected node $v_{ME}^{p^*}$ depends on the Shannon Entropy,

$$v_{ME}^{p^*} = \underset{\forall i=\{1,..,n\} \wedge Q(i)=False}{\operatorname{argmax}} - \sum_{v_a^q \in \Sigma_v^q} P\left[v_i^p, v_a^q\right] log\left(P\left[v_i^p, v_a^q\right]\right) \tag{3.12}$$

Computational cost: $O(n^2)$

3.3.2.2 Query by committee

The general idea of this strategy would be to use several graph matching algorithms and then use the obtained correspondence as models. Due to the huge amount of time that would involve, we will leave the study of how to efficiently implement this method as a future work.

We define two different strategies based on *Expected Model Change*: *Maximum Gradient Norm* and *Maximum Probability Change given a Common Labelling*.

3.3.2.3 Expected Model Change

We define two strategies.

3.3.2.3.1 Maximum Gradient Norm (MGN)

The magnitude of the training gradient of variable $P\left[v_i^p, v_a^q\right]$ is defined as $M\left(v_i^p, v_a^q\right)$ (section 3.1). The learner should query the node $v_{MGN}^{p^*}$ defined through the following equation,

$$v_{MGN}^{p^*} = \underset{\forall i=\{1,..,n\} \wedge Q(i)=False}{\text{argmax}} \left\{ \sum_{v_a^2 \in \Sigma_v^2} P\left[v_i^p, v_a^q\right] M\left(v_i^p, v_a^q\right) \right\} \quad \textbf{(3.13)}$$

Computational cost: $O(n^4)$, considering that computing M is $O(n^2)$

3.3.2.3.2 Maximum Probability Change given a Common Labelling (MPCCL)

The learner should query the instance that if its current correspondence is changed, this would result a maximum increase in its probability,

$$v_{MPCCL}^{p^*} = \underset{\forall i=\{1,..,n\} \wedge Q(i)=False}{\text{argmax}} \left\{ \underset{\forall a=\{1,..,n\}}{max} \left\{P\left[v_i^p, v_a^q\right]\right\} - P\left[v_i^p, f\left(v_i^p\right)\right] \right\} \quad \textbf{(3.14)}$$

If $\underset{\forall j=\{1,..,n\}}{max} \left\{P\left[v_i^p, v_a^q\right]\right\} > P\left[v_i^p, f\left(v_i^p\right)\right]$, the current mapping of v_i^p is not the ideal one, considering only probabilities $P\left[v_i^p, v_a^q\right]$, for all $v_a^q \in \Sigma_v^q$. On the contrary, if $\underset{\forall a=\{1,..,n\}}{max} \left\{P\left[v_i^p, v_a^q\right]\right\} = P\left[v_i^p, f\left(v_i^p\right)\right]$, then, the current mapping is the one that obtains the maximum probability, therefore, it is the ideal case.

Computational cost: $O(n^2)$.

3.3.2.4 Variance Reduction & Estimated Error Reduction

These methods are not presented in this thesis. Both methods are based on statistical analysis. In our case, we only have one instance per class. For this reason, it seems difficult to be applied in our domain.

3.3.2.5 Density-Weighted methods

It is not trivial to decide which node can be considered as an outlier simply considering the graph. For this reason, there are not considered this type of methods in this thesis.

3.3.3 Active & Interactive graph matching algorithm

In this section it is presented the *Active and Interactive Graph Matching* algorithm that obtains a correspondence between nodes of *attributed graphs* G^p and G^q considering the oracle feedback. The algorithm computes several times a sub-optimal graph matching algorithm $Graph\_Matching$ (for instance [32, 33, 30]), but in each step, the cost matrices C_v and C_e are updated through the current feedback (section 3.3.1). However, as commented on section 3.1, we need the probability matrix P to be accessible. This matrix is usually initialised depending on the edit costs or simply taking values $1/n$. The cost matrices C_v and C_e are application dependent and they are initialised through function $Initialise\_Cost$. The algorithm finishes when the feedback of the user is OK or all the nodes of G^p have been queried, $Q(i) = True$ for all nodes of G^p. When the active algorithm is initialised, $Q\left(v_i^p\right)$ takes the $False$ value for all nodes of G^p and this value is changed to $True$ in each query. The final *matching cost* obtained at the end of the algorithm is computed through the original costs, C_v^0 and C_e^0. This is because the aim of modifying these costs is to influence the correspondence but not to change the resulting cost given a correspondence. Function $Active\_Query$ returns a selected node that it is supposed to be the most informative for the system (section 3.3.2). Besides, function $Oracle\_Feedback$ returns the set of simple actions proposed by the user (section 3.3.1). We call k the number of the user's simple actions in each iteration and hk the total number of simple actions. Function $Append$ appends the new actions in the set w into the historical set hw. Finally, actions $Interactive\_Node\_Costs$ and $Interactive\_Edge\_Costs$ updates the costs matrices (section 3.3.1). These functions are detailed above. The input of these two last functions is not w but hw. The reason is twofold. On one hand, we have realised that some graph matching algorithms modify the costs matrices and we want these matrices to be the original ones in each step, except the costs imposed by the interactive module. On the other hand, we need the history of all impositions to be considered in the edge case. Note that the algorithm does not control if the user imposes contradictory orders.

Figure 3.4 shows the probabilistic graph matching framework with interactive and active learning. Dashed lines connect the active modules that do not appear in the classical framework shown in Figure 1.4. Moreover, we have added the original costs C_v^0 and C_e^0, since the *matching cost* C_f is computed through these costs.

```
Algorithm       Active & Interactive Graph Matching
Input:          Attributed Graphs $G^p$ and $G^q$
Output:         Correspondence $f$ and Cost $C_f$
Pseudocode:     $hk = 0$  // total number of actions
                $C_v^0, C_e^0 = Initialise\_Cost(G^p, G^q, k_v, k_e)$; $C_v = C_v^0$; $C_e = C_e^0$.
                $P = Initialise\_Probabilities(C_v, C_e)$.
                $(P, f) = Graph\_Matching(P, C_v, C_e)$.
                Do
                        $v^p = Active\_Query(P, f)$.
                        $(w, k) = Oracle\_Feedback(G^p, G^q, v^{p^*}, f, \ldots)$.
                        $hw = Append(hw, w)$.  $hk = hk + k$.
                        $C_v = Interactive\_Node\_Costs(hw, C_v, hk)$.
                        $C_e = Interactive\_Edge\_Costs(hw, C_e, hk)$.
                        $(P, f) = Graph\_Matching(P, C_v, C_e)$.
                Since
                Stop
                Compute $C_f(C_v^0, C_e^0)$
End
Algorithm
```

Algorithm 3.1. Active & Interactive graph matching.

```
Function        Interactive Node Costs
Input:          Historical feedback $hw$
                Cost Matrix $C_v$
                Total number of actions $hk$
Output:         Cost Matrix $C_v$
Pseudocode:     Do      $\forall m = \{1, \ldots, hk\}$
                        //being $hw^{(m)} = Set(v_i^p, v_a^q)$
                        $C_v[v_i^p, v_a^q] = 0$
                        $C_v[v_i^p, v_b^q] = \infty \ \forall b \neq a$
                        $C_v[v_j^p, v_a^q] = \infty \ \forall j \neq i$
End Function
```

Algorithm 3.2. Interactive Node Costs.

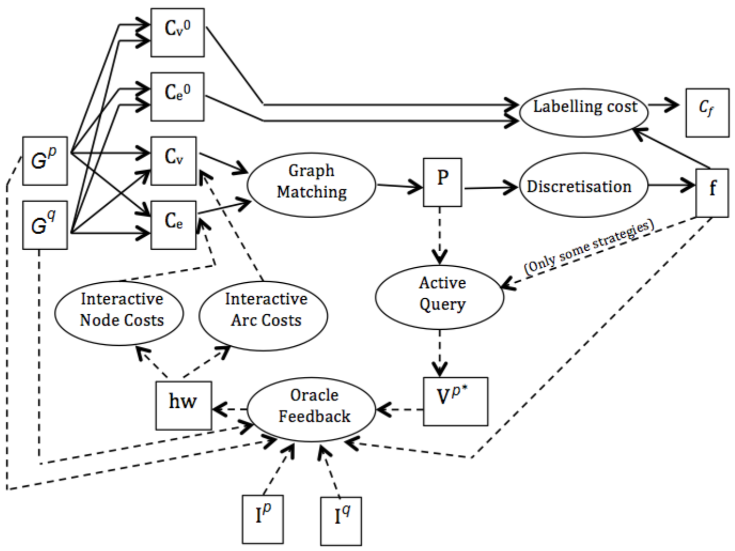


Figure 3.4. Probabilistic graph matching framework with interactive and active learning.

```
Function        Interactive Edge Costs
Input:          Historical feedback $hw$
                Cost Matrix $C_e$
                Total number of actions $hk$
Output:         Cost Matrix $C_e$
Pseudocode:     Do    $\forall m1, m2 = \{1, \dots, hk\}, m1 < m2$
                      //being  $hw^{(m1)} = Set(v_i^p, v_a^q) \wedge$
                               $hw^{(m2)} = Set(v_j^p, v_b^q)$
                      $C_e[v_i^p, v_a^q, v_j^p, v_b^q] = 0 \wedge C_e[v_j^p, v_b^q, v_i^p, v_a^q] = 0$
                      $C_e[v_i^p, v_{a'}^q, v_j^p, v_{b'}^q] = \infty \wedge C_e[v_j^p, v_{b'}^q, v_i^p, v_{a'}^q] = \infty$ $\forall a \neq$
                      $a'$ and $b \neq b'$
                      $C_e[v_{i'}^p, v_a^q, v_{j'}^p, v_b^q] = \infty \wedge C_e[v_{j'}^p, v_b^q, v_{i'}^p, v_a^q] = \infty$
                      $\forall i \neq i'$ and $j \neq j'$

End Function
```

Algorithm 3.3. Interactive Edge Costs.

The input of the *Oracle_Feedback* algorithm may contain extra information (like the images I^p and I^q) from which the graphs have been extracted. This is because the oracle, may need to see these extra information to propose a correspondence.

3.4 Experimental Evaluation

To experimentally validate our method, we have used the well-known graph databases: *LETTER-HIGH*, *COIL-DEL*, *GREC* [72] described in the Databases annex of this thesis. These databases have different characteristics such as cardinality, diversity, mean number of nodes and so on. Nevertheless, we considered only the (x, y) attribute on the nodes and edges have no attributes. Several graph matching algorithms and graph-class prototypes have been compared using these datasets [123]. We have not taken into consideration the separation of these graphs into classes in any of the experiments. We have also used *HOUSE* and *HOTEL* datasets [81] also described in the Databases annex. The first four rows of Table 3.1 show the main database characteristics.

From each database, we have defined a test set as follows. The Test Set is composed of T elements $\{G^t, G'^t, \hat{f}^t\}$ that have a pairs of graphs and a correspondence: $\{\{G^1, G'^1, \hat{f}^1\} \dots \{G^t, G'^t, \hat{f}^t\} \dots \{G^T, G'^T, \hat{f}^T\}\}$. Graphs G^t, $1 \leq t \leq T$, are the ones in the original databases. T is the number of graphs of these datasets. Graphs G'^t, $1 \leq t \leq T$, have been randomly generated through distorting G^t. The attribute value of nodes has been modified as follows. We add a Gaussian random value (fifth and sixth row of Table 3.1) to the original attribute value. Edges have been removed or added following the probability shown in the last row of Table 3.1. Both graphs G^t and G'^t have the same number of nodes. The correspondence $\hat{f}^t$ has been computed together with the generation of the distorted graph G'^t and we assume it is the best correspondence between G^t and G'^t. Nevertheless, by construction, we do not guarantee this correspondence to obtain the minimum cost. We make the oracle's feedbacks coincide with these correspondences. That is, we assume that the feedback at the G^{th} iteration of the test $\{G^t, G'^t, \hat{f}^t\}$ is $w^{t(m)} = Set\left(v^{*t(m)}, \hat{f}^t\left(v^{*t(m)}\right)\right)$ where $v^{*t(m)}$ is the node of G^t selected by the active method at the m^{th} iteration. This is because we consider that the oracle feedback is composed of only one simple action and that it is related to the selected node by the active module.

We have used *Graduated Assignment* [32] as the graph matching algorithm, which is based on probabilities. Other algorithms could be used such as

Expectation Maximization [33] or *Probabilistic Relaxation* [30], but we assume they have no influence on the relative differences between the active queries precision results, for these databases.

Database	***LETTER HIGH***	***COIL DEL***	***GREC***	***HOUSE-HOTEL***
# Nodes	4.7	21.5	11.5	30
# Edges	4.5	54.2	12.2	38.1
Mean node value (x and y)	1.6	60.7	272,9	318.4
St. Dev. node value	1	31.6	139,8	104.5
Mean noise on node	0	0	0	0
St. Dev. noise on node value	0.75	15	1-20	10
Probability on edge noise	0.04	0.04	0.025-0.4	0. 04

Table 3.1. Database characteristics.

In this practical evaluation we show three different measures: *node precision* and *learning curve* and *matching cost* throughout the iterations.

We want an active method to suggest a node $v^{*t(m)}$ of G^t that the current mapping obtained by the graph matching algorithm $f^{t(m)}$ is different from the oracle's mapping $\hat{f}^t$. This is because, in this case, the active and interactive methods will help the system to improve its output since the oracle will try to correct the wrong mappings. We are not interested in the cases where $f^{t(m)}\left(v^{*t(m)}\right) = \hat{f}^t\left(v^{*t(m)}\right)$ since the oracle's interaction is going to be useless. Given an element of the test set $\left\{G^t, G'^t, \hat{f}^t\right\}$ and the obtained mapping $f^{t(m)}$ in the G^{th} iteration, the delta function represents the usefulness of the active and interactive actions as follows,

$$\delta\left(w^{t(m)}, f^{t(m)}\right) = \begin{cases} 1 & \text{if } \; w^{t(m)} = Set\left(v^{*t(m)}, v'\right) \text{ and } f^{t(m)}(v^*) \neq v' \\ 0 & \qquad\qquad otherwise \end{cases} \tag{3.15}$$

If the active method has been useful then $\delta\left(w^{t(m)}, f^{t(m)}\right) = 1$. Note that, due to the construction of our experiments, the delta function can be rewritten as,

$$\delta\left(v^{*t(m)}, \hat{f}^t, f^{t(m)}\right) = \begin{cases} 1 & \text{if } \; f^{t(m)}\left(v^{*t(m)}\right) \neq \hat{f}^t\left(v^{*t(m)}\right) \\ 0 & \qquad otherwise \end{cases} \tag{3.16}$$

The *Node Precision* of an active method is computed as the average delta function of all the elements $\{G^t, G'^t, \hat{f}^t\}$ of the test set at the first iteration. The first iteration is only used because the costs on nodes or edges could be different throughout the active techniques in the other ones.

$$Node\ Precision = \frac{\sum_{t=1}^{T} \delta\left(v^{*t(1)}, \hat{f}^t, f^{t(1)}\right)}{T} \quad (3.17)$$

3.4.1 Noise Robustness Study

Figure 3.5 shows the *precision* obtained at the first interaction of all strategies and also a random strategy on *GREC* database with different noise levels. In the random strategy, the selected node is obtained randomly. The *precision* on all strategies is better than the random strategy, although MPCCL clearly obtains the lower values and MGN the best ones. When noise level increases, the graph matching algorithm mismatches more node mappings, and then the *precision* tends to increase because the probability of selecting a non-correct mapped node increases (3.17). Clearly, when there is a high level noise, all strategies obtain similar results to the random one.

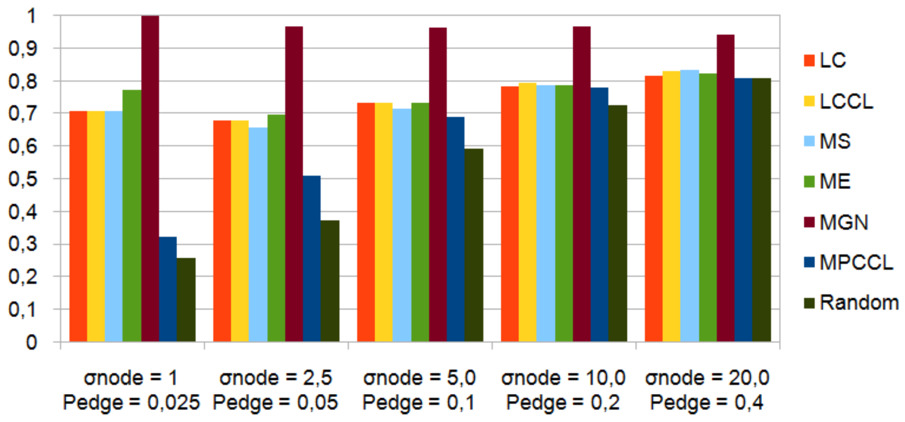


Figure 3.5. Precision at first iteration of *GREC* database throughout different noise levels.

The *learning curve* is obtained through the *hamming distance* between the current correspondence and the oracle's correspondence of all the elements

$\{G^t, G'^t, \hat{f}^t\}$ of the test set at each iteration *m*th. This is because we want the current correspondence to be as similar as possible to the oracle's correspondence.

3.4.2 Node Precision and Learning Curve Study

Figure 3.6 shows the average *node precision*. MPCCL strategy obtains the worst results and LC, LCCL, ME and MS obtain similar results. If the oracle's correspondence does not obtain the minimum cost, in other words, it is not the optimal correspondence, MGN obtains bad results. This is because this method has a strong dependency on the cost function (more precisely, the gradient of the cost function). In *LETTER* and *COIL* datasets, the relative noise added to the original graphs is high (note the relation between the original values of node attributes and the added noise in Table 3.1), and so, the oracle's correspondence is not usually the optimal one, in other words, it does not obtain the minimum cost. This is not the case with the *HOUSE* and *HOTEL* datasets, in which the relative noise is not so high.

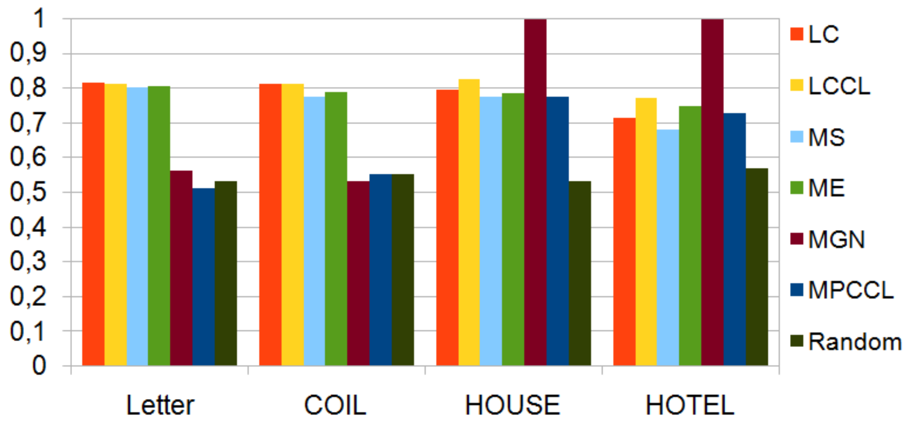
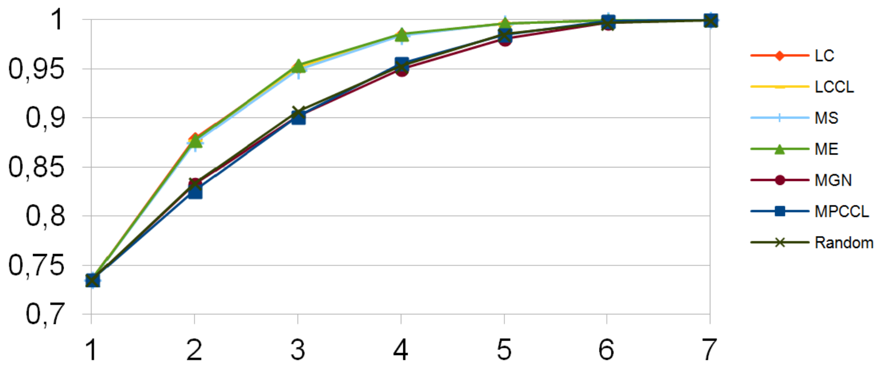


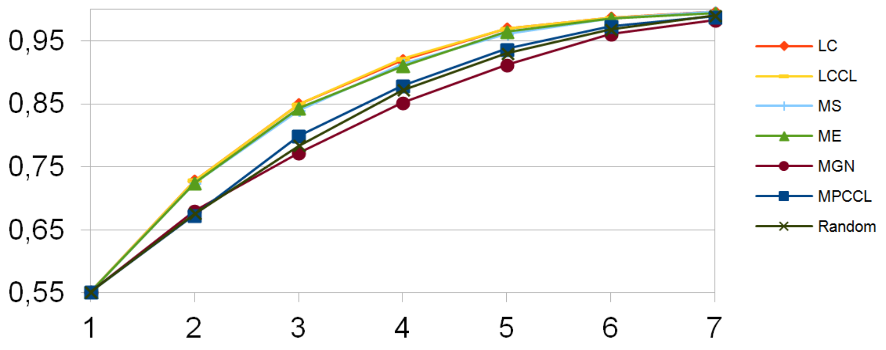
Figure 3.6. Precision at first iteration of some datasets applying the commented active methods and a random method.

The *learning curve* is obtained through the *hamming distance* equation (1.1) between the current correspondence and the oracle's correspondence of all the elements $\{G^t, G'^t, \hat{f}^t\}$ of the test set at each iteration q. This is because we want the current correspondence to be as similar as possible to the oracle's correspondence.

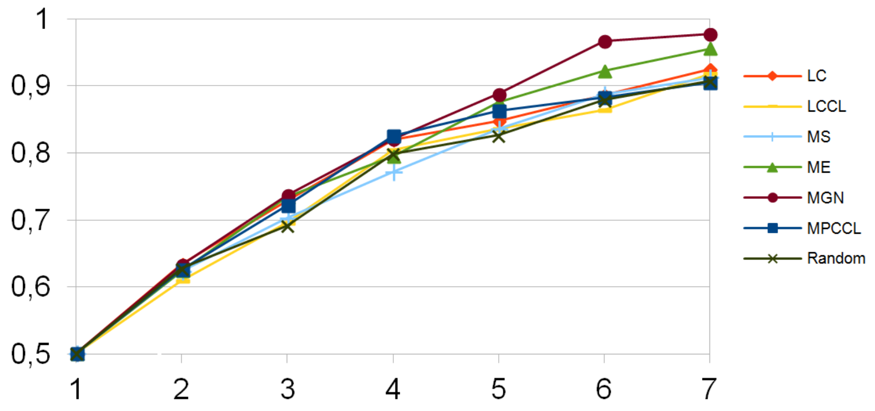
$$LeraningCurve^{(q)} = \sum_{t=1}^{T} \frac{n^t - \Delta^H\left(\hat{f}^t, f^{t(m)}\right)}{n^t} \tag{3.18}$$



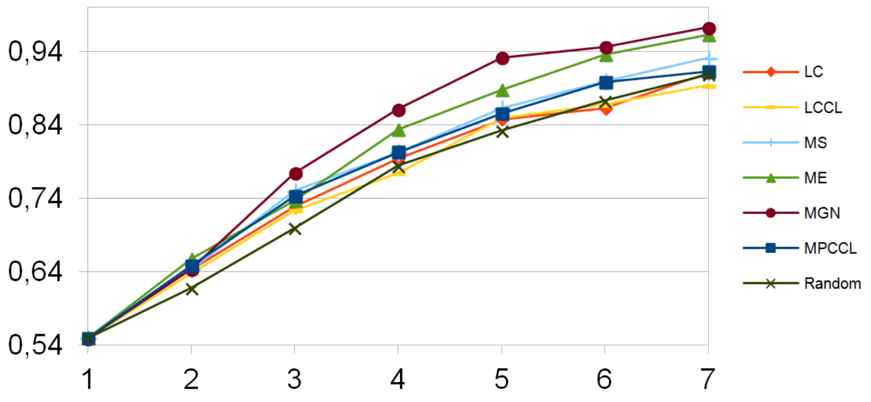
a. *Learning curve for LETTER-HIGH dataset. LC, LCCL, MS & ME obtain similar results. MGN, MPCCL and Random obtain similar results.*



b. *Learning curve for COIL-DEL dataset. LC, LCCL, MS & ME obtain similar results.*



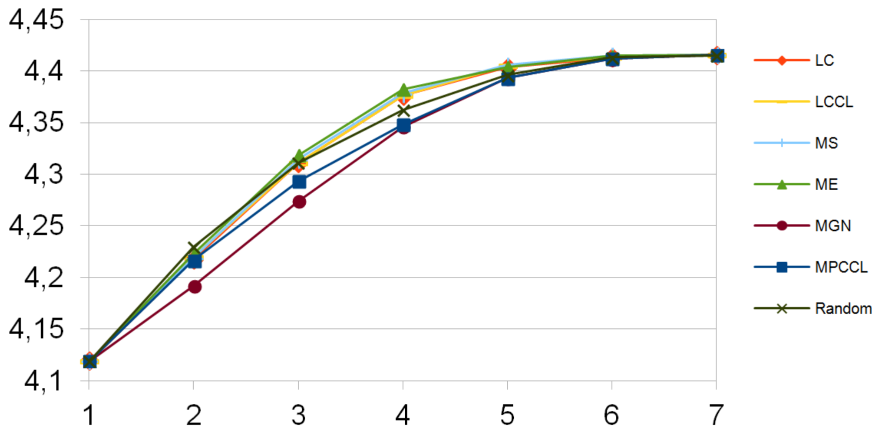
c. *Learning curve for HOUSE dataset.*



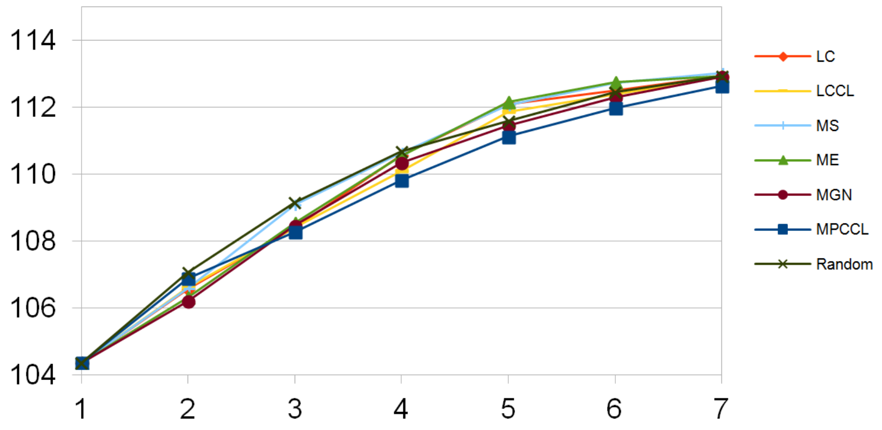
d. *Learning curve for HOTEL dataset.*

Figure 3.7. Learning curves versus number of iterations of different active strategies.

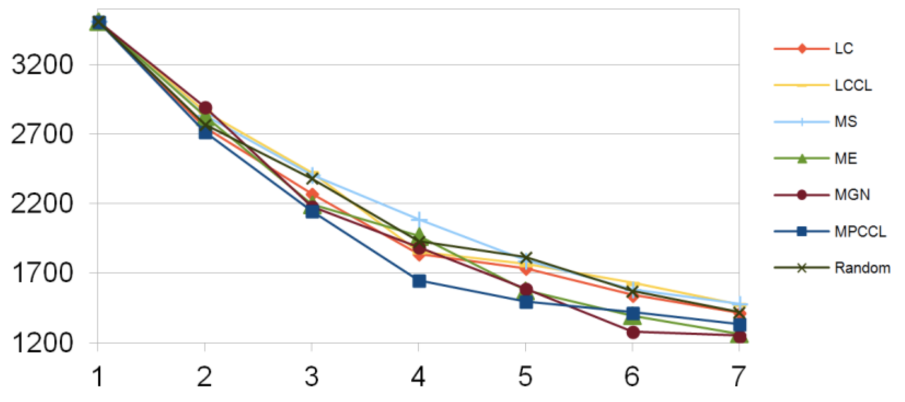
Figure 3.7 shows the *learning curve* for all databases applying the explained strategies. In *LETTER* and *COIL* datasets (figures Figure 3.7.a and Figure 3.7.b), the best results appear in the *Uncertain Sampling* strategies (MS, LC, LCCL & ME). The *Expected Model Change* strategies (MPCCL & MGN) are not useful because they obtain similar results to random strategy. Conversely, ME obtains the best results in *HOUSE* and *HOTEL* datasets (figure Figure 3.7.c and Figure 3.7.d). Note that there is a clear relation between the *node precision* (Figure 3.6) and the *learning curve* (Figure 3.7). If the strategy selects the nodes that have not been properly matched, then the system learns faster. Since, in a real application, we cannot compute the *learning curve* because we do not have, a priori, the complete oracle's correspondence, we can use the *node precision* to decide which strategy to use.



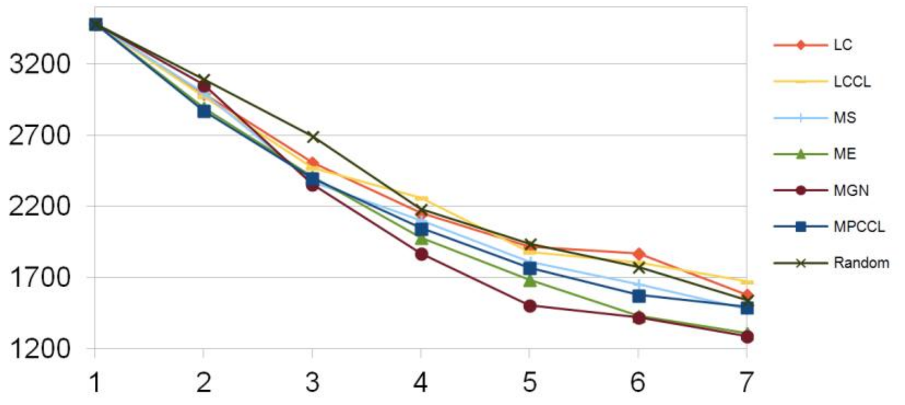
a. *Average cost curve for LETTER-HIGH dataset. LC, LCCL, MS & ME obtain similar results.*



b. *Average cost curve for COIL-DEL dataset.*



c. *Average cost curve for HOUSE dataset.*



d. *Average cost curve for HOTEL dataset.*

Figure 3.8. Average curves versus number of iterations of different active strategies.

Finally, we show in Figure 3.8 the normalised average cost of the current correspondences throughout iterations. If the cost tends to decrease (*HOUSE* and *HOTEL* datasets), then the representation of objects through the attribute graphs is in accordance with the oracle's feedback and the oracle improves the graph matching algorithm performance. In other words, the oracle proposes optimal correspondences. Conversely, if the cost tends to increase (*LETTER*s and *COIL* datasets), the oracle's feedback is not in accordance of the representation of the objects and the distance measure. We have to consider that the increase or decrease of the cost is not related to the quality of the active method but to the relation between the representation of the data and the oracle's point of view of the involved objects. In our examples, the noise added to the graphs in the *LETTER* and *COIL* datasets is larger than in the *HOUSE* and *HOTEL* datasets. Therefore, in most of the cases, the oracle's correspondence does not obtain the minimum cost, and so the cost, while the system learns, increases instead of decreases.

$$Cost^{(q)} = \frac{\sum_{t=1}^{T} C_{f^{t(m)}}(G^t, G'^t)}{T} \tag{3.19}$$

Chapter 4
Learning Substitution Weights for Error-Tolerant Graph Matching

In Chapter 3 it has been presented a new framework to add interactivity to *Error-Tolerant Graph Matching* aiming to obtain the perfect correspondence after a few interactions. But this framework does not perform any learning strategy taking advantage of the knowledge provided by the oracle.

The next two chapters explore the feasibility of learning the parameters for *Graph Edit Distance* to minimize the *hamming distance* between the optimal correspondence (the least cost correspondence) and the ground-truth one (provided by an oracle) using the *Edit Cost Error* as a loss function. The present chapter is focused on learning the substitution weights for the nodes attributes and Chapter 5 is focused on learning the insertions and deletions costs.

4.1 Matching graphs with weighted attributes

A particular way of defining a distance between nodes for *Graph Edit Distance* is weighting its attributes to control which attribute is more important.

Given two *attributed graphs* G^p and G^q (formally defined in 1.2) and a node assignment (unary assignment) $v_i^p \to v_a^q$, the weighted distance between both graph nodes $d(v_i^p \to v_a^q) \in \mathbb{R}^t$ is defined through any distance measure on their attribute values $d(v_i^p \to v_a^q) = distance\left(\gamma^p\left(v_i^p\right), \gamma^q\left(v_a^q\right)\right)$ and $g(e_{ij}^p \to e_{ab}^q) = 0$ if both edges exist or both edges not exist or $g(e_{ij}^p \to e_{ab}^q) = 1$ otherwise. Following these definitions, we define the *Local distance* of two graphs G^p and G^q given a bijective mapping represented by the correspondences matrix F (defined in 1.2.1.1) as a vector of $t + 1$ elements,

$$\Phi(G^p, G^q, f) = \left[\sum_{ia=1}^{n+m} d(v_i^p \to v_a^q)F_{ia}\,, \sum_{iabj=1}^{n+m} g(e_{ij}^p \to e_{ab}^q)F_{ia}F_{bj}\right] \qquad \textbf{(4.1)}$$

We define the *weighted cost* C_w of matching two graphs G^p and G^q given the correspondences f as an inner product,

$$C_w(G^p, G^q, f) = \langle \Phi(G^p, G^q, f), w \rangle \tag{4.2}$$

Where $w \in \mathbb{R}^{t+1}$ is a vector of weights $w = [w_1, \dots, w_t, w_{t+1}]$ that weights each of the attributes on the nodes $[w_1, \dots, w_t]$ and the importance of the edges $[w_{t+1}]$[4]. The aim of these weights is to gauge the importance of each of the attributes and they are application or database dependent. If we consider w is an input parameter on the domain of function $C_w(G^p, G^q, f)$ and we consider to be constants both graphs G^p, G^q and the correspondence f; then C_w is an increasing hyperplane such that when $w = \vec{0}$ it happens that $C_w = 0$ due to $\Phi(G^p, G^q, f) \geq \vec{0}$ (see [35] for more information).

Finally, given two graphs G^p and G^q and a vector of weights w, the generic formulation of the graph matching problem with weighted attributes consists of finding the optimal correspondence $\dot{f}_w$ given by the solution of the assignment problem.

$$\dot{f}_w = \underset{f}{\operatorname{argmin}}\, C_w(G^p, G^q, f) \tag{4.3}$$

Note that there are several optimal matching matrices $\dot{f}_w$ depending on the value of the weights w. And given a specific w, the matching $\dot{f}_w$ cannot be unique [35].

4.1.1 An example of graph matching with weighted attributes

Suppose we have two complete graphs with order 4. Nodes have two attributes that represent their bi-dimensional position (X, Y) and for simplicity we consider that the weight of the edges always is 1. Graph G^p has the following node attributes: $v_1^p = (1{,}4)$, $v_2^p = (2{,}3)$, $v_3^p = (4{,}1)$, $v_4^p = (2.5{,}2.5)$. Graph G^q has the following node attributes: $v_1^q = (1{,}2)$, $v_1^q = (2{,}1)$, $v_1^q = (4{,}4)$, $v_1^q = (1.5{,}1.5)$. Figure 4.2 shows 8 different optimal correspondences between graphs of Figure 4.1 (as commented, the optimal correspondence depends on the weights w). Nodes of G^p are represented by blue squares whereas nodes of G^q are represented by red rhombus. Note

[4] w can be extended to weight attributed edges.

that $\dot{f}_{[w_x, w_y]}$ represents an optimal correspondence when the weights vector is $w = [w_x, w_y]$. In the next section, we show that each of these eight pairs of weights (for instance $w = [25,5]$) is located in a different correspondence region.

We also show in Table 4.1 the obtained local distance and the Cost of each correspondence given the specific weights. In this example, we have defined $d_{ia} = \left[\left(\gamma_x^p(v_i^p) - \gamma_x^q(v_a^q)\right)^2, \left(\gamma_y^p(v_i^p) - \gamma_y^q(v_a^q)\right)^2\right]$. Moreover, in order to simplify the example, the graph is complete and so the structural information does not influence on the final distance and optimal node correspondence.

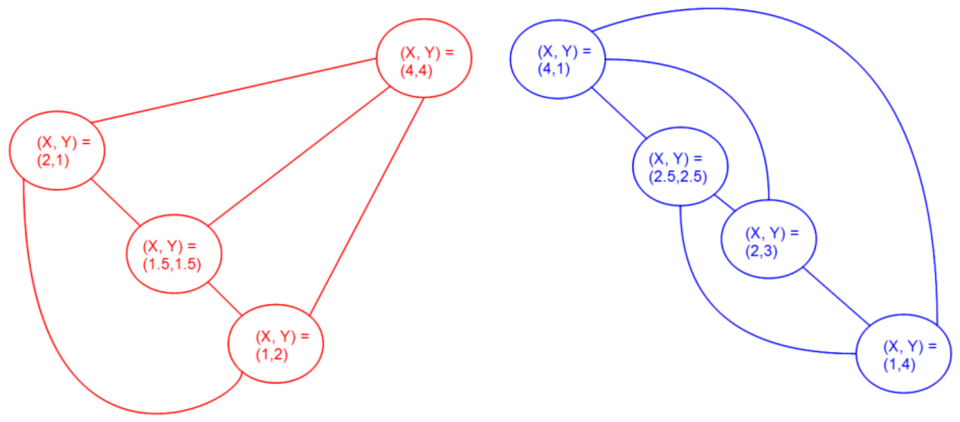
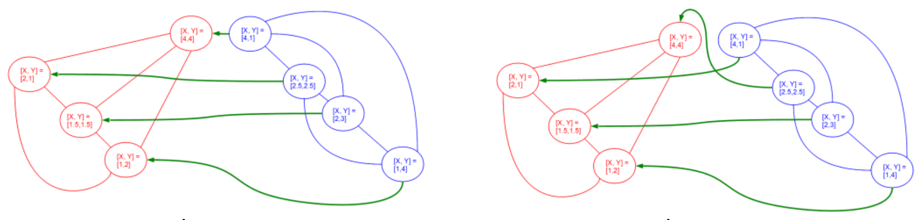


Figure 4.1. Two graphs where the circles represent the nodes (X and Y are the attributes) and the lines between them represents the edges.



$f^1 = \dot{f}_{[w_x=25, w_y=5]}$ $f^2 = \dot{f}_{[w_x=25, w_y=20]}$

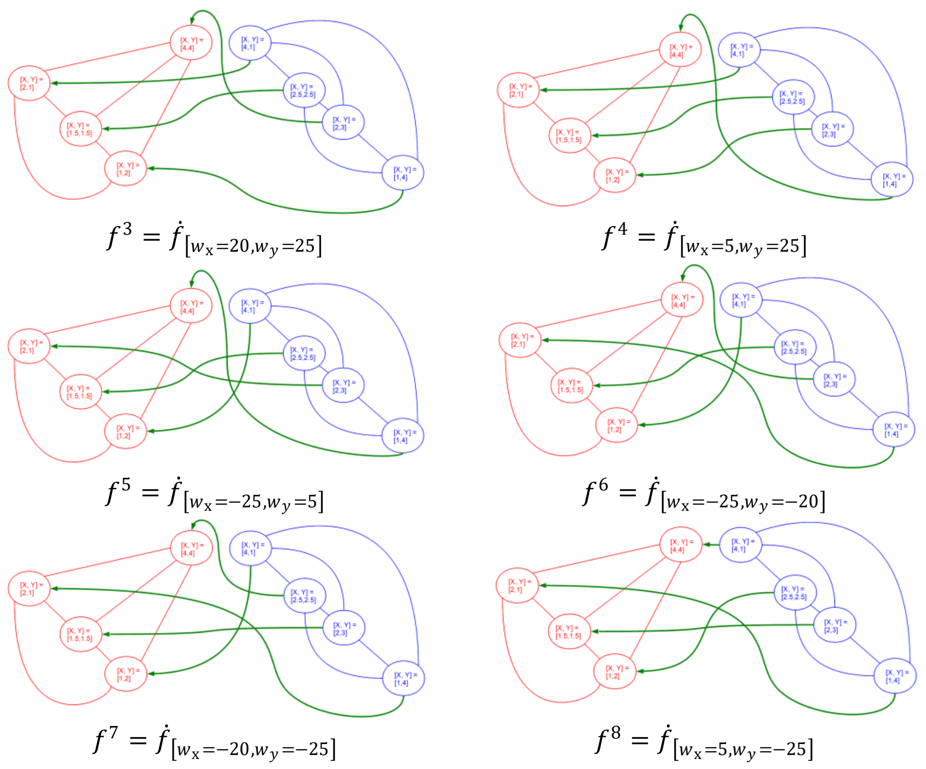

[X, Y] = [4,4]
[X, Y] = [2,1]
[X, Y] = [1.5,1.5]
[X, Y] = [1,2]
[X, Y] = [4,1]
[X, Y] = [2.5,2.5]
[X, Y] = [2,3]
[X, Y] = [1,4]
$f^3 = \dot{f}_{[w_x=20, w_y=25]}$
$f^4 = \dot{f}_{[w_x=5, w_y=25]}$
$f^5 = \dot{f}_{[w_x=-25, w_y=5]}$
$f^6 = \dot{f}_{[w_x=-25, w_y=-20]}$
$f^7 = \dot{f}_{[w_x=-20, w_y=-25]}$
$f^8 = \dot{f}_{[w_x=5, w_y=-25]}$

Figure 4.2. Two graphs and eight optimal correspondences $\dot{f}_w$ (green arrows) given different weights w.

$[w_x, w_y]$	$\Phi\left(G^p, G^q, \dot{f}_{[w_x,w_y]}\right)$	$C_w\left(G^p, G^q, \dot{f}_{[w_x,w_y]}\right)$	$\dot{f}_{[w_x,w_y]}$
[25,5]	[0.5,17.5]	100	f^1
[25,20]	[6.5,8.5]	332.5	f^2
[20,25]	[20,25]	330	f^3
[5,25]	[15,2]	125	f^4
[−25,5]	[19,6]	-445	f^5
[−25, −20]	[15,12]	-615	f^6
[−20, −25]	[12.5,14.5]	-612.5	f^7
[5, −25]	[3.5,20.5]	-495	f^8

Table 4.1. Local distance and Costs of the optimal correspondence given eight different weights combinations. For instance, in first row, $25 \cdot 0.5 + 5 \cdot 17.5 = 100$.

The local distance Φ defined in equation (4.1) is always positive and the cost C_w defined in equation (4.2) is positive or negative depending on the weights. Due to the graph matching problem has been defined such as the optimal correspondence is the one that minimises C_w (equation (4.3)), if negative weights are imposed, then it means that we wish to make the distance between the attributes as larger as possible, so, we want these attributes to be different.

4.2 Correspondences Space

The *Correspondences Space* (first defined in [35]) is a Euclidean space where the coordinates correspond to the graph edit insertion and deletion costs. In our case, each coordinate represents an element of the weights vector w. We can deduce some regions in this space, called *Class of Costs* [35], such that all points in each region of the *Correspondence Space* obtain the same optimal correspondence $\dot{f}_w$.

Property 1: A *Class of Costs* is a region on the *Correspondence Space* that forms a radial sector that has its origin at $w = \vec{0}$. The hyperplanes that limit the radial sector are points such that more than one optimal correspondence achieves the same cost C_w.

Demo: Given two *Classes of Costs*, f' and f'', the border of these *Classes of Costs* is composed of points in the *Correspondence Space* such that both Classes have the same cost C_w given a value of w. That is, $\langle \Phi(G^p, G^q, f'), w \rangle = \langle \Phi(G^p, G^q, f''), w \rangle$ for all values w.

Then $\langle \left(\Phi(G^p, G^q, f') - \Phi(G^p, G^q, f'') \right), w \rangle = 0$, which is a first order equation system in which variables are the elements in vector w and so the solution is an hyperplane. If $w = \vec{0}$ then the equation holds for all correspondences and for this reason all borders of *Classes of Costs* converge to $\vec{0}$. Moreover, it is obvious that $\underset{f}{\operatorname{argmin}}\{C_w(G^p, G^q, f)\} = \underset{f}{\operatorname{argmin}}\{\tau \cdot C_w(G^p, G^q, f)\}$ and so $\dot{f}_w = \underset{f}{\operatorname{argmin}}\{\langle \Phi(G^p, G^q, f), \tau \cdot w \rangle\}$. Thus, all combinations of variables w have to keep the same proportion■

Figure 4.3 shows the *Class of Costs* on the *Correspondence Space* of two graphs in Figure 4.2. We can see that, given any combination of weights, there are only 8 *Classes of Costs* and so, 8 optimal correspondences which are the ones shown in Figure 4.2. Note that there are $4! = 24$ possible correspondences but only 8 become optimal considering any combination of weights.

Moreover, given the *Correspondence Space* and two graphs G^p and G^q, we can define a function defined through the *Correspondence Space* that its value in each point is the cost function given the optimal correspondence $C_w\left(G^p, G^q, \dot{f}_w\right)$. This function is called $EditSurface$ [35].

$$EditSurface_{G^p,G^q}(w) = C_w\left(G^p, G^q, \dot{f}_w\right) \qquad \textbf{(4.4)}$$

The aim of this definition is simply to show that we want the domain of the function to be the *Correspondence Space* instead of both graphs that are considered constants. Contrarily, the cost function assumes that the weights are constant and previously validated.

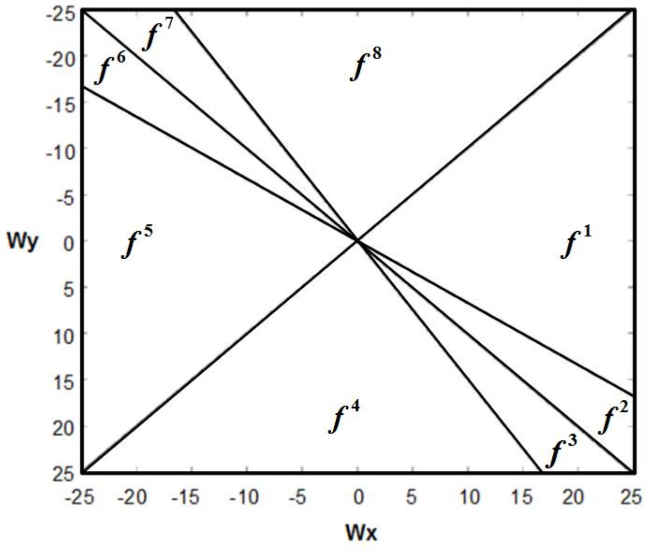
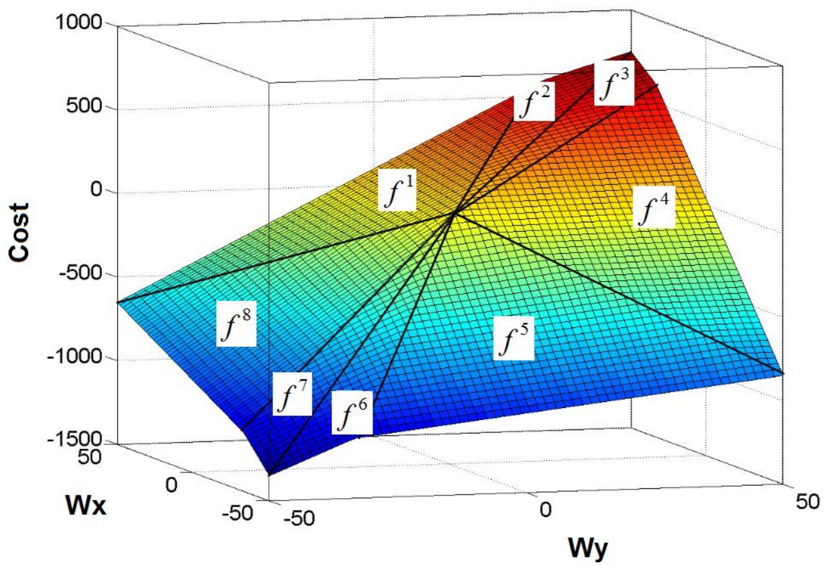
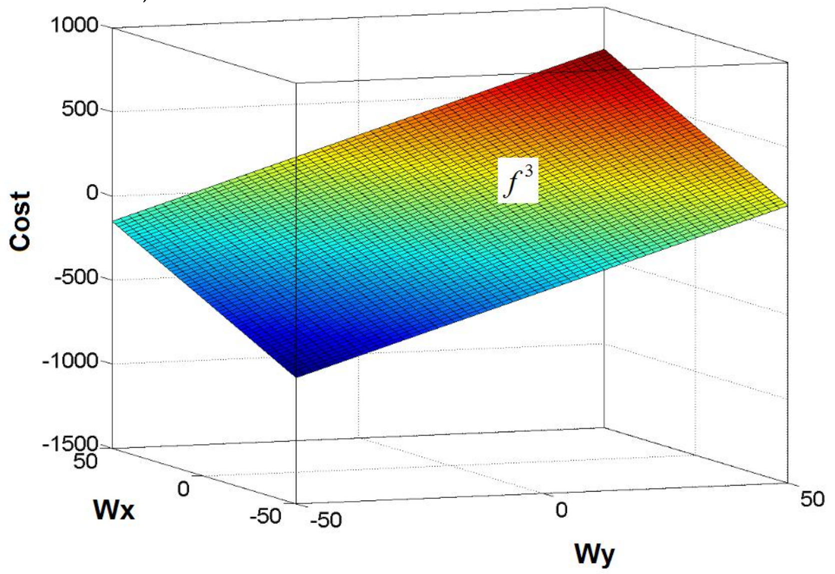


Figure 4.3. The 8 Classes of Costs of our example.



a. *$EditSurface_{G^p,G^q}(w)$ of graphs in Figure 4.2.*



b. *Cost of correspondence f^3, $C_w(G^p, G^q, f^3)$.*

Figure 4.4. Costs surfaces.

Figure 4.4.a shows the *EditSurface* obtained from the example in Figure 4.2. It is an increasing function composed of 8 planes where $EditSurface_{G^p,G^q}(\vec{0}) = 0$ and at (0,0) the eight planes coincide. Note that the domain of each plane is a *Class of Costs* (Figure 4.3).

Moreover, the borders of these *Classes of Costs* coincide on the non-derivable parts of the $EditSurface$. If we consider a specific correspondence, for instance f^3, we see that it is only optimal in a small region (Figure 4.4.a) in which $C_w(G^p, G^q, f^3)$ (Figure 4.4.b) obtains the minimum cost.

4.3 Learning graph matching

Similarly to [39], we approach the problem of learning the weights w for graph matching as a supervised learning problem [141]. The training set is composed of N observations. Each observation is composed of a pair of graphs $\left(G^{p^n}, G^{q^n}\right)$. Moreover, the label (or class) of each pair of graphs is the matching $\hat{f}^n$ (this correspondence has to be seen as the ideal correspondence for an oracle, the ground-truth correspondence). Note that the optimal correspondence $\dot{f}_w^n$ given a pair of graphs $\left(G^{p^n}, G^{q^n}\right)$ depends on the weights w (equation (4.3)) but the ideal correspondence matrix $\hat{f}^n$ does not depend on them. For this reason, the aim of learning graph matching, we purpose, is to search for the weights w such that $\dot{f}_w^n$ tends to be as close as possible to $\hat{f}^n$ given the training set of the N pairs of graphs $\left(G^{p^n}, G^{q^n}\right)$ and the ideal set of correspondences matrices $\hat{f}^n$.

We use the standard approach of minimising the empirical risk plus a regularisation term weighted by a parameter λ. The empirical risk is the average loss in the training set since the prediction of the loss on the test set is assumed not to be available. The optimal predictor will then be the one that minimises the following expression,

$$\frac{1}{N}\sum_{n=1}^{N} \Delta_w^C\left(G^{p^n}, G^{q^n}, \hat{f}^n, \dot{f}_w^n\right) + \lambda \cdot \Omega(w) \qquad \textbf{(4.5)}$$

Where $\Delta\left(G^{p^n}, G^{q^n}, \hat{f}^n, \dot{f}_w^n\right)$ is the loss incurred by the assignment problem when predicting $\dot{f}_w^n$ instead of $\hat{f}^n$ for training input $\left(G^{p^n}, G^{q^n}\right)$. The regularisation term $\Omega(w)$ penalises solutions of w that are far away from the problem at hand.

In the following, we focus on setting up the optimisation problem by presenting three different options for the loss function $\Delta \in [0,1]$ and the regularisation term $\Omega \in [0, \infty)$.

4.3.1 Minimising the hamming distance

In fact, the main aim of the learning process at hand is to reduce to the minimum *hamming distance* (see equation (1.1)) between the automatic obtained correspondences and the ground-truth ones. But this loss function has two main problems. Firstly, it is not continuous on the border of the correspondence regions. Secondly, inside these regions, the function is constant. These two properties make the function not appropriate for the optimisation algorithms. In the following, we show the function that we propose to overcome these problems.

Figure 4.5 shows Δ^H defined in section 1.2.1.2 when the ideal correspondence is f^3 (see Figure 4.2).

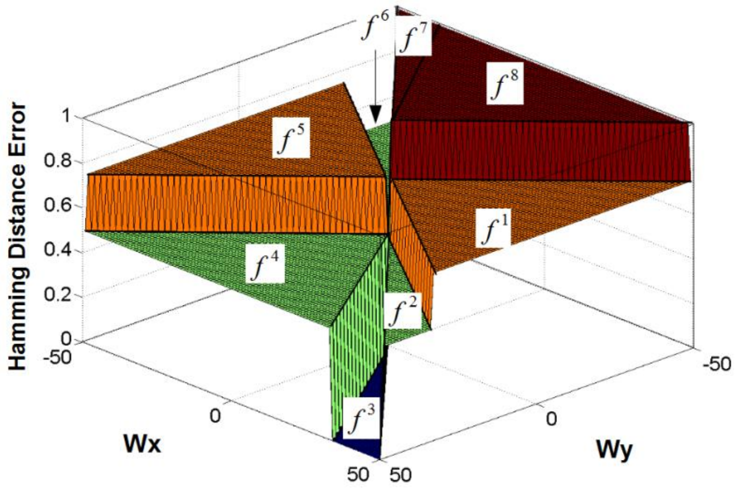


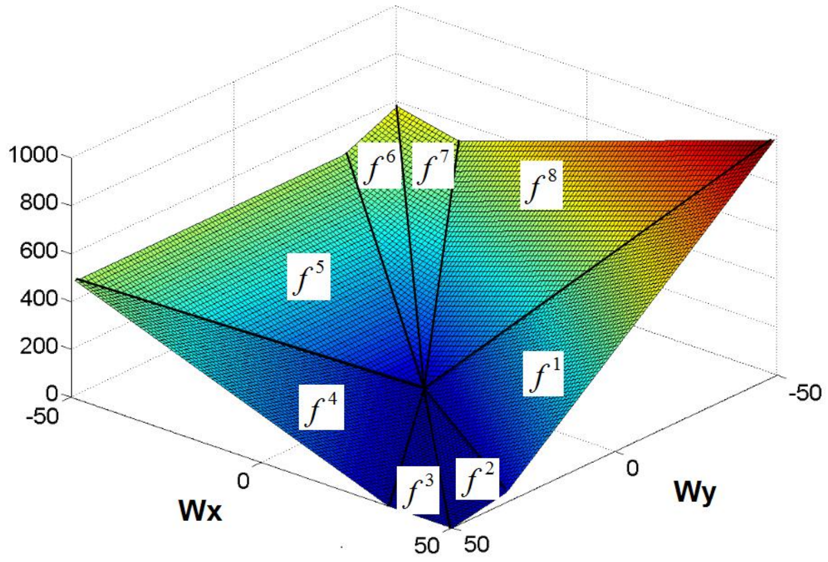
Figure 4.5. Hamming distance when the ideal correspondence is f^3 in Figure 4.2.

4.3.2 Edit Cost Error

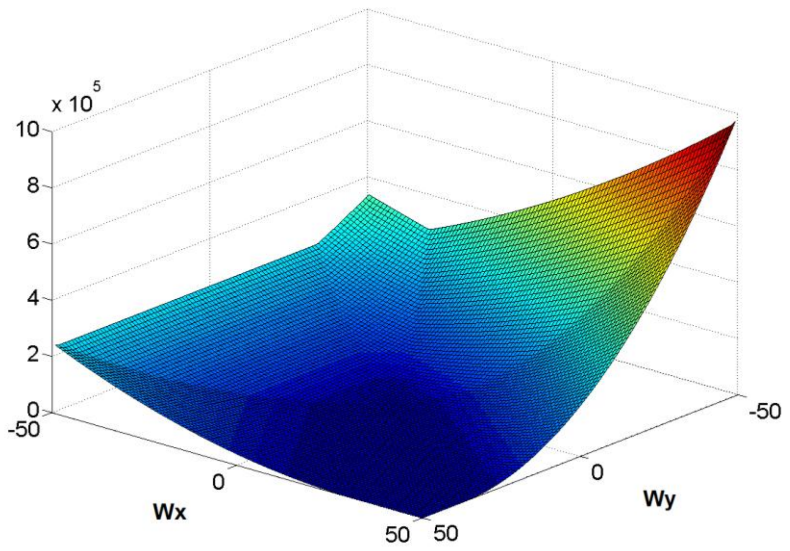
We propose the loss function called *Edit Cost Error*, Δ_w^C. It gauges how far we are to obtain the cost generated by the ideal correspondence matrix.

$$\Delta_w^C\left(G^{p^n}, G^{q^n}, \hat{f}^n, \dot{f}_w^n\right) = \left(C_w\left(G^{p^n}, G^{q^n}, \hat{f}^n\right) - C_w\left(G^{p^n}, G^{q^n}, \dot{f}_w^n\right)\right)^2 \qquad \textbf{(4.6)}$$

We assume that two correspondences that are close to each other tend to achieve similar costs C_w. Although this relation is not true for all graphs and correspondences, the empirical evaluation shows that it is true for most of the graphs and correspondences in the datasets. Moreover, in cases where there are symmetries on the graphs, correspondences $\hat{f}^n$ and $\dot{f}_w^n$ can be completely different but represent a similar mapping between graph nodes. In these cases, our evaluation function obtains better results than the *hamming distance* function.



a. *Without the quadratic term.*



b. *With the quadratic term.*

Figure 4.6. Cost Error Δ_w^C when the ideal correspondence is f^3.

Property 2: a) IF $w = \vec{0}$ then $\Delta_{\vec{0}}^C\left(\hat{f}, \dot{f}_{\vec{0}}\right) = 0$.

b) $\Delta_w^C\left(G^p, G^q, \hat{f}, \dot{f}_w\right)$ is an increasing function such that $\Delta_w^C\left(G^p, G^q, \hat{f}, \dot{f}_w\right) \geq 0$.

Demo: a) It is trivial since $\langle Vector, \vec{0} \rangle = 0$

b) By definition of $\dot{f}_w$ we have $C_w\left(G^p, G^q, \hat{f}\right) \geq C_w\left(G^p, G^q, \dot{f}_w\right)$ and so $\Delta_w^C\left(G^p, G^q, \hat{f}, \dot{f}_w\right) \geq 0$. Besides, given a specific correspondence h and two weight vectors $w^{'}$ and $w^{''}$ such that $\|w^{'}\| \geq \|w^{''}\|$ then $\Delta_{w^{'}}^C\left(G^p, G^q, \hat{f}, h\right) \geq \Delta_{w^{''}}^C\left(G^p, G^q, \hat{f}, h\right)$. This is because $\Delta_{w^{'}}^C\left(G^p, G^q, \hat{f}, h\right) = \langle V, w^{'} \rangle \geq \langle V, w^{''} \rangle = \Delta_{w^{''}}^C\left(G^p, G^q, \hat{f}, h\right)$ where $V = \left(\Phi\left(G^p, G^q, \hat{f}\right) - \Phi(G^p, G^q, h)\right)$. The inequality $\langle V, w^{'} \rangle \geq \langle V, w^{''} \rangle$ holds since $\langle V, w^{'} \rangle \geq 0$ ■

Figure 4.6 shows Δ_w^C when the ideal correspondence is f^3 obtained through equation (4.6).

4.3.3 Regularization term

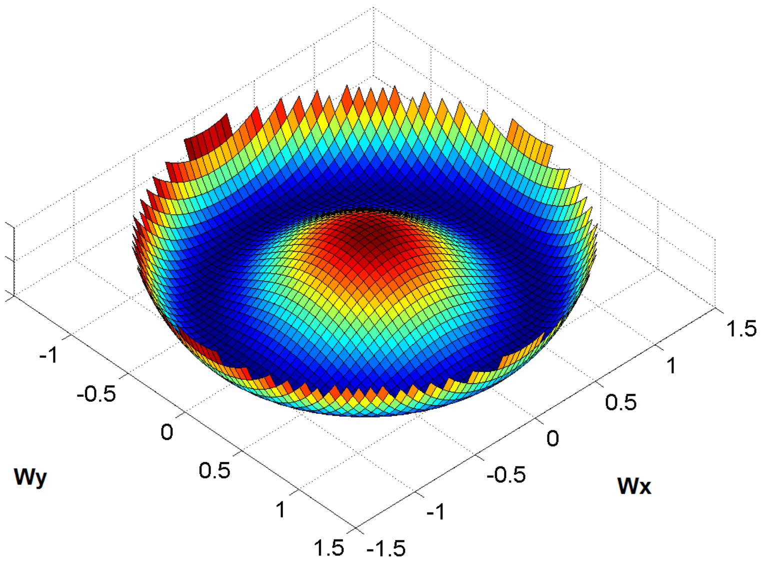


Figure 4.7. Regularisation term. $\lambda \cdot \Omega_w^c$. When $\lambda = 1$.

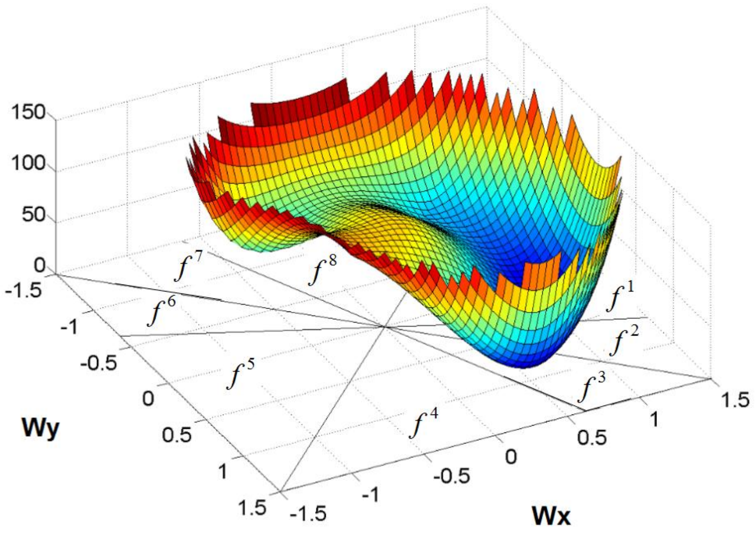


Figure 4.8. Optimal predictor $\Delta_w^C + 1 \cdot \Omega^C(w)$.

Due to properties 1 and 2, Δ^C minimises at $w = \vec{0}$. But this is a solution that all correspondences $\dot{f}_w$ obtain $\Delta^C_{\vec{0}}\left(G^p, G^q, \hat{f}, \dot{f}_w\right) = 0$ independently of how close are to the ground-truth correspondence $\hat{f}$. For this reason, we propose the following regularisation term:

$$\Omega^c_w = (\|w\|^2 - 1)^2 \tag{4.7}$$

Figure 4.8 shows loss function (4.8), which is our proposal of optimal predictor with the same data than in Figure 4.6 when the ideal correspondence is f^3. Clearly, there are the same local minima.

$$Predictor_w = \Delta^C_w + \lambda \cdot \Omega^c_w \tag{4.8}$$

4.3.4 The algorithm

Algorithm 4.1 schematically shows our optimisation algorithm. The algorithm seeks for the vector of weights $\dot{w}$ that obtains the minimum value of $Predictor_w$ through any quadratic programming algorithm. For instance, *trust-region* [142], *levenberg-marquardt* [143] or *nelder–mead* [143]. Moreover the computation of Δ^C_w can be done by any quadratic [30, 32, 82, 33, 25] or linear [69] assignment problem.

```
Algorithm      Class of Costs Predictor
Define:        $\dot{f}^n_w = \underset{f}{\operatorname{argmin}}\, C_w(G^{p^n}, G^{q^n}, f)$
               $Predictor(w) = \frac{1}{N}\sum_{n=1}^{N} \Delta^C_w\left(G^{p^n}, G^{q^n}, \hat{f}^n, \dot{f}^n_w\right) + \lambda \cdot \Omega^c_w$
Input:         $N$ pairs of graphs $(G^{p^n}, G^{q^n})$ and correspondence
               matrices $\hat{f}^n$
Output:        A vector of optimal weights $\dot{w}$
Pseudocode:    $\dot{w} = \underset{w}{\operatorname{argmin}}\{Predictor(w)\}$
End
Algorithm
```

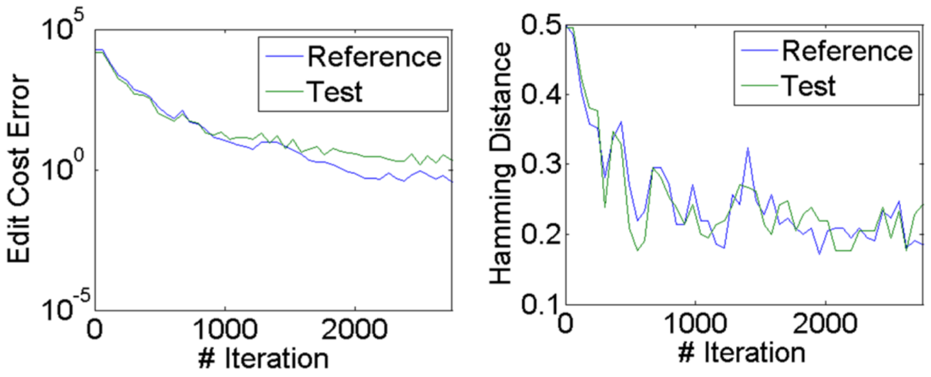
Algorithm 4.1. Active & Interactive graph matching.

Optimization algorithms [142, 143, 143] need an initialization of weights. Due to, there is no aprioristic information of the best initialization, in the validation step, several initializations could be evaluated. We have used the initialization $w = \vec{1}$ because we assume in our datasets, the attributes positively contribute in the matching cost (attributes are intended to benefit the correspondence) and also we assure $\|w\|^2 > 1$.

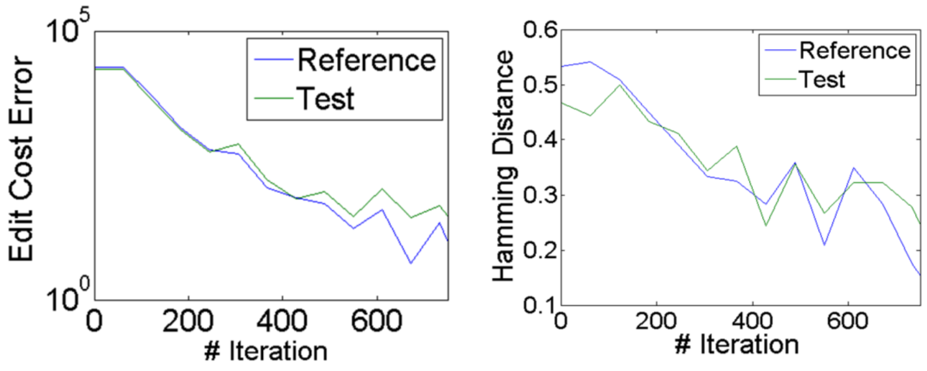
4.4 Experimental evaluation

We split the practical evaluation in two sections. In the first one, we empirically validate the assumption that the lower is the *Edit Cost Error*, the lower is the *hamming distance*. We have made a comparative study with respect to the results presented in [39]. They used the *Bundle method* [145] (as the optimisation algorithm) and the *Graduated Assignment algorithm* [32] (for the quadratic assignment). It has been used the *Trust-region* [142] (as the optimisation algorithm) and the *Fast Bipartite Graph Matching* [67, 29] (for the quadratic assignment). *Trust-region* was selected because it has an implementation in the MATLAB native libraries (we used default parameters). In some initial experiments, we also used the *Nelder–Mead* method [143] and we obtained similar results. We decided $\lambda = 1$ through a validation process. We validated our method with the same two databases used to validate the method we want to compare with [39]. We have used *HOUSE*, *HOTEL* and *HORSE* databases described in the Databases annex.

4.4.1 Correlation between the Edit Cost Error and the Hamming Distance



a. *HOUSE*

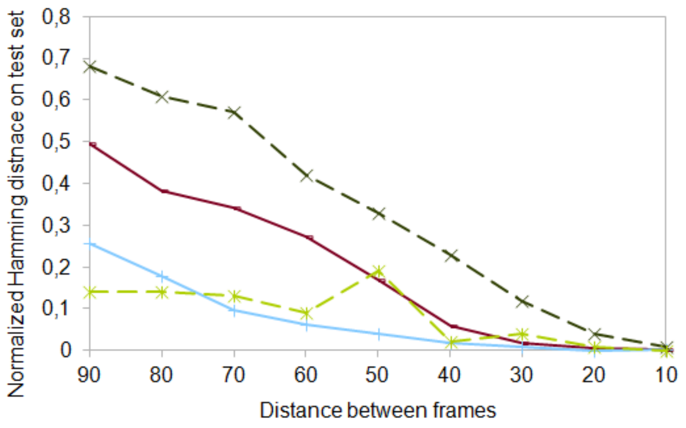


b. *HOTEL*

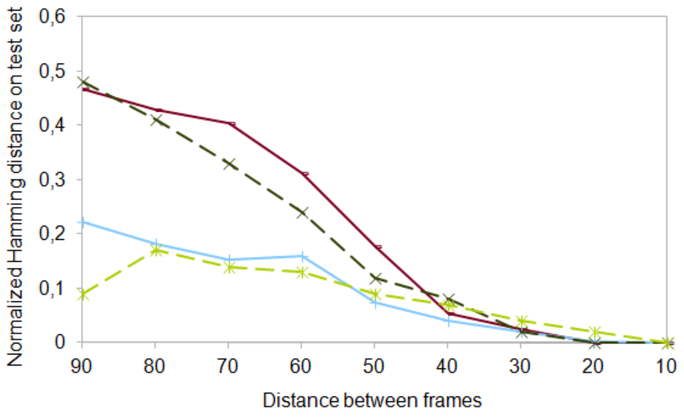
Figure 4.9. (left) Edit Cost Error and (right) hamming distance with respect to the number of iterations.

Figure 4.9 shows the evolution (with respect to iterations of the learning algorithm) of the *Edit Cost Error* and the *Hamming Distance* between the obtained correspondence and the ground-truth correspondence in the reference and test graphs of the *HOUSE* and *HOTEL* sequences. We realise that in both sets and in both functions, there is a general decrease in the metric value. Therefore, we empirically validate that making decrease the *Edit Cost Error* forces to decrease the *hamming distance*.

4.4.2 Graph matching accuracy



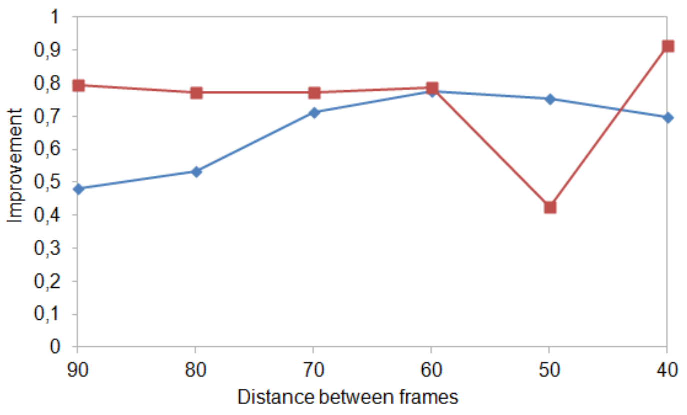
a. *HOUSE*



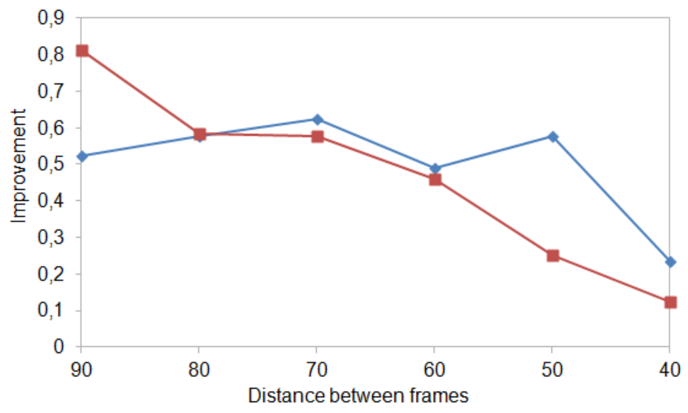
b. *HOTEL*

Figure 4.10. Normalized hamming distance on the HOUSE and HOTEL sequences with respect to different frame separations (in [39], $\boldsymbol{\lambda^E = 0.1}$). ━ our method without learning, ─┼─ our method with learning, ─×─ results presented in [39] without learning, ─✳─ results presented in [39] with learning.

The minimisation of the *hamming distance* between the automatically extracted correspondence and a possible ground-truth correspondence is at the core of most of classification, clustering or machine learning in general applications based on graphs. For this reason, the aim of the following experiments is to show the obtained *hamming distance* between the automatically extracted correspondence and the ground-truth correspondence. We show some tests extracted from [39] and other ones obtained using our method. We not only show the *hamming distance* but the *hamming distance* improvement of the learned weights with respect to the initial weights, to compare the results with [39].



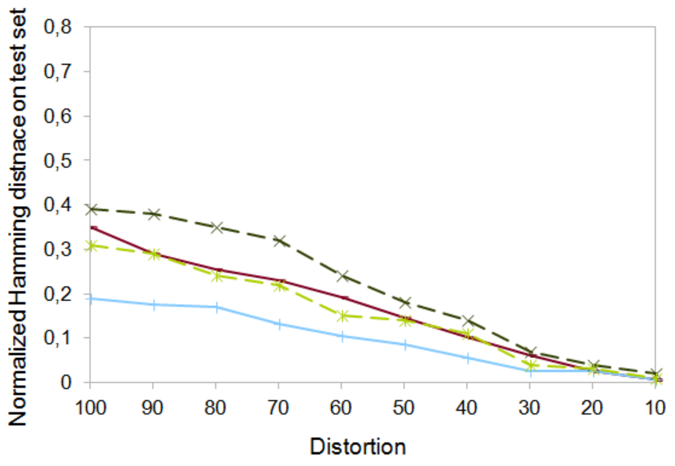
a. *HOUSE*



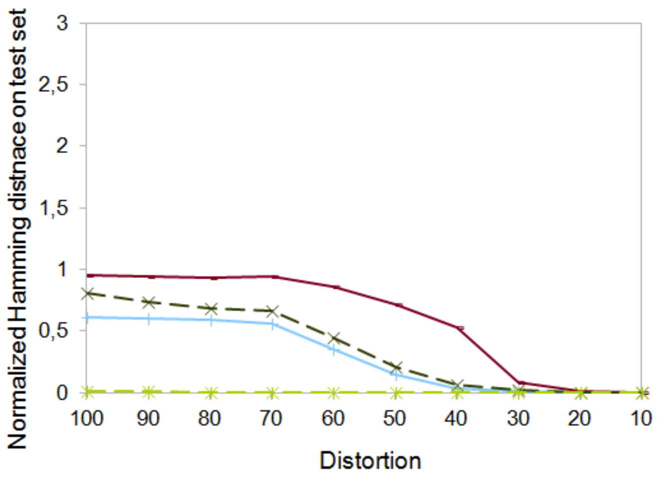
b. *HOTEL*

Figure 4.11. ─◆─ our improvement, ─■─improvement presented in [39] on the HOUSE and HOTEL sequences with respect to different frame separations.

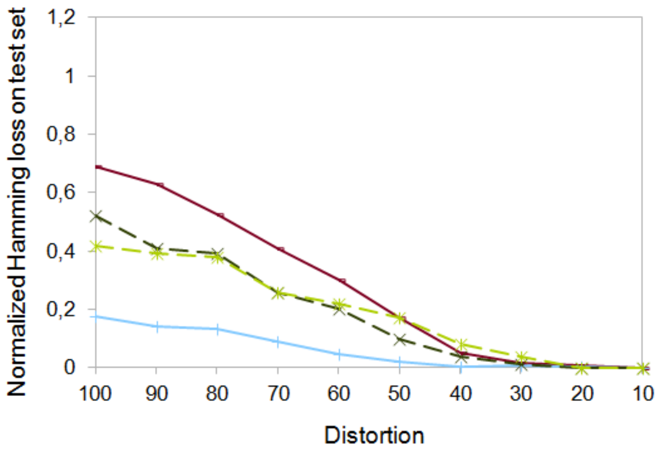
Figure 4.10 and Figure 4.11 shows the average *hamming distance* (normalised by the number of nodes) between the automatically extracted correspondence and the ground-truth correspondence using the *HOUSE* and *HOTEL* sequences. The horizontal axis is the distance in frames of the compared images (in fact, with respect to the extracted graphs from these images). As it is logical, in the whole experiments, the tests with learned costs outperform their corresponding tests without learning. In some cases we obtain better results than the ones in [39].



a. *HORSE-NOISE*



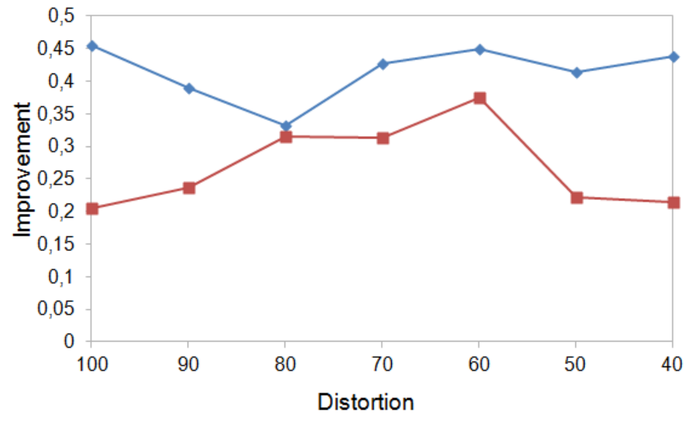
b. *HORSE-ROTATE*



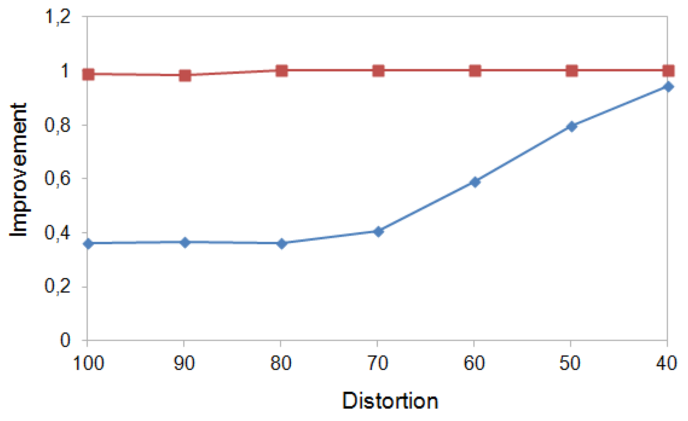
c. *HORSE-SHEAR*

Figure 4.12. Normalized hamming distance on the HORSE database: NOISE, ROTATE and SHEAR sequences with respect to different frame separations. ━ our method without learning, ─┼─ our method with learning, ─×─ results presented in [39] without learning, ─✳─ results presented in [39] with learning.

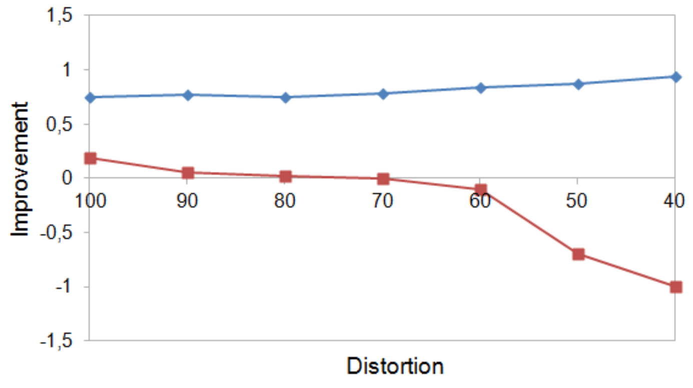
Figure 4.12 shows the *hamming distance* (normalised by the number of nodes) and Figure 4.13 the improvement obtained with our method and the method in [8]. We obtain a higher improvement in *HORSE-NOISE* and *HORSE-SHEAR* databases. In the case of *HORSE-ROTATE* database, although we improve the *hamming distance*, the other method obtains better results due to it achieves a *hamming distance* of almost zero with the learned weights. Finally, note that our improvement is always positive but it is not the case of their method in the *HORSE-SHEAR* database.



a. *HORSE-NOISE*



b. *HORSE-ROTATE*



c. *HORSE-SHEAR*

Figure 4.13. ◆ our improvement, ■ improvement presented in [39] on the HORSE database: NOISE, ROTATE and SHEAR sequences with respect to different frame separations.

Figure 4.14 shows the 61 learned weights in *HOUSE* sequence.

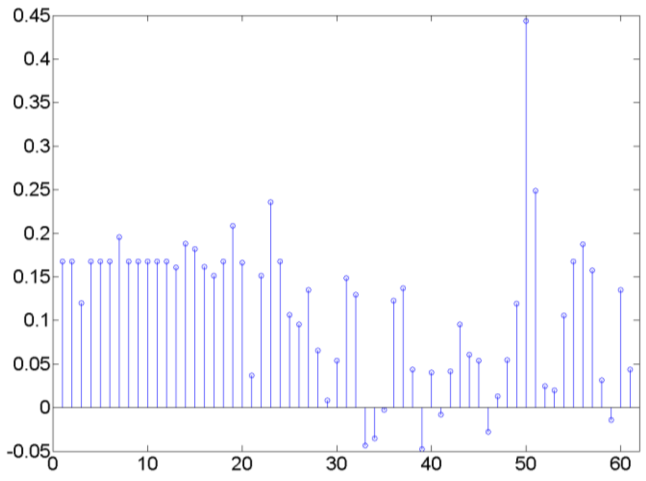


Figure 4.14. Weights we learned on the HOUSE sequence (baseline 90).

We note that the performance depends on the database but the average result with our method is slightly higher than [39].

Chapter 5
Learning Edit Costs for Error-Tolerant Graph Matching

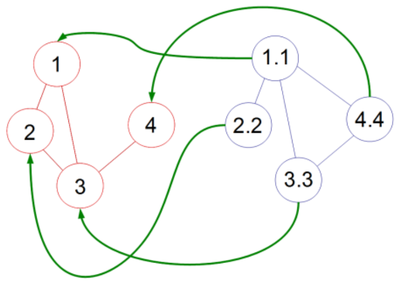
The distortion of the *Graph Edit Distance* is defined through edit operations: insertion, deletion or substitution of nodes and edges. To quantitatively evaluate the degree of distortion, a penalty cost to each edit operation is defined according to the amount of distortion that it introduces in the transformation. Although a proper definition of these costs is a cornerstone of classification or clustering applications, little research has been done to automatically find them. Usually, they are established through a manual validation process.

In the Chapter 4 it has been shown how to learn the substitution weights for nodes attributes such that *hamming distance* between the automatic and the ground-truth correspondence is minimised. The aim of the present chapter is to demonstrate the applicability of a loss function similar to *Edit Cost Error* presented in Chapter 4 when calculating the penalty cost of each edit operation (edit costs).

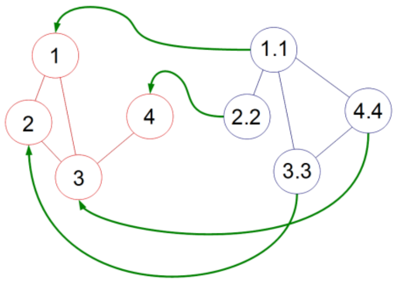
5.1 On the influence of the Insertion and Deletion Edit Costs on the Hamming Distance

As introduced in Chapter 1, the *Graph Edit Distance* is defined as the minimum amount of required distortion to transform one graph into the other through inserting, deleting and substituting nodes and edges. Moreover, some penalty costs are assigned to each edit operation. Deletion/Insertion operations are transformed to assignations in f of non-null nodes of the first/second graph to null nodes of the second/first graph. Depending of this edit cost (K_v and K_e), a different optimal correspondence is achieved.

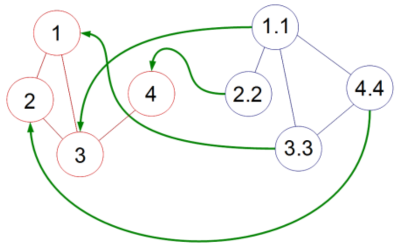
Figure 5.1 shows the whole optimal correspondences that can be achieved through all possible combinations of $[K_v\,, K_e]$ between two sample graphs. The domain of all of these combinations is called the *Correspondences Space*. In the case of the two involved graphs of this example, there are exactly 11 optimal correspondences. The *Correspondences Space* (see 4.2 for more information) has an important role in our optimization method.



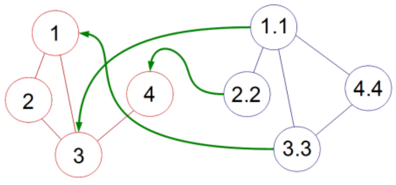
f^1



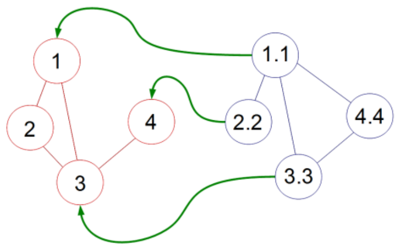
f^2



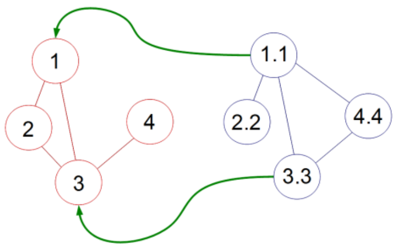
f^3



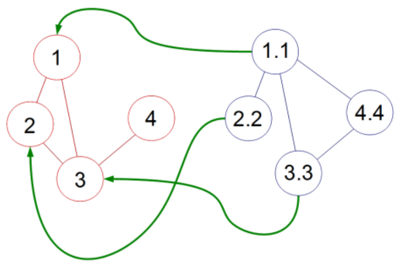
f^4



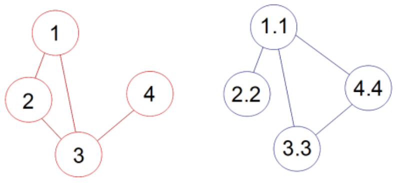
f^5



f^6



f^7



f^8

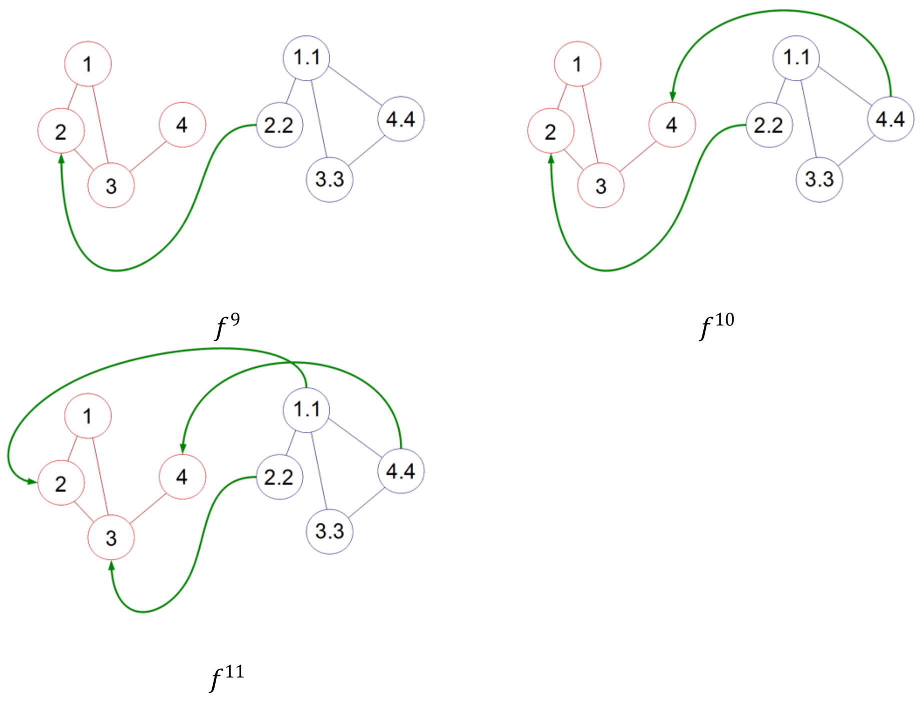


Figure 5.1. The 11 optimal correspondences (green arrows) throughout the Correspondences Space given a pair of graphs.

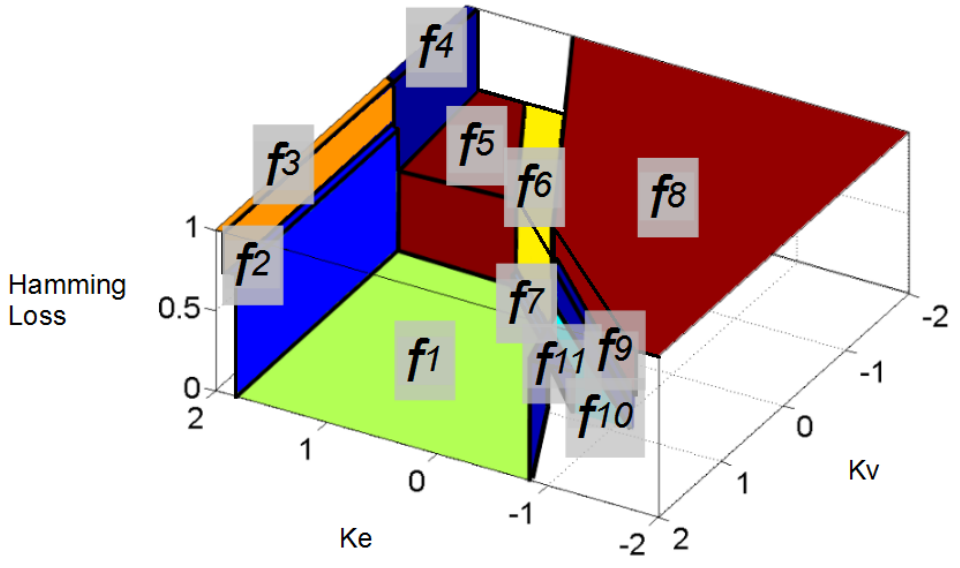


Figure 5.2. Hamming distance (normalised by the order of the graphs) between f^1 and the other correspondences.

Figure 5.2 shows the *hamming distance* (normalised by the number of nodes, $n = 4$) between the first correspondence, f^1, and the other ones. Each correspondence defines a flat region. Note that the $\mathrm{d_H}(f^1, f^1) = 0$.

Suppose a human specialist looks at two images and the *attributed graphs* extracted from these images and deduce a correspondence $\hat{f}$ (for instance two X-ray images in which some local parts have been extracted such as bones). If we wish to automatize the process, we desire the automatically extracted correspondence $\dot{f}$ to be as closer as possible to the correspondence deduced by the human, $\hat{f}$. The aim of this chapter is to define a learning model for compute costs K_v and K_e such that the *hamming distance* between the automatically obtained correspondence $\dot{f}$ and the specialist's correspondence $\hat{f}$ is minimised.

5.2 Learning K_v and $K_e$costs

The predictor is the one that minimises the following expression,

$$Predictor_{[K_v,K_e]} = \Delta^C_{[K_v,K_e]} + \lambda \cdot \Omega^c_{[K_v,K_e]} \tag{5.1}$$

Where $\Delta_{[\mathrm{K_v},\mathrm{K_e}]}(G^{p^n}, G^{q^n}, \hat{f}^n, \dot{f}^n)$ is the loss incurred by the assignment problem when predicting $\dot{f}^n$ instead of $\hat{f}^n$ given a training input (G^{p^n}, G^{q^n}) and a pair of costs K_v and K_e.

Given all combinations of costs K_v and K_e, Figure 5.3 and Figure 5.4 show the *Edit Cost* of correspondence f^1 and the optimal correspondence $\dot{f}$ respectively.

Finally, Figure 5.5 shows the *Edit Cost Error* (equation (5.2)) where the ground-truth correspondence $\dot{f}$ is f^1.

$$\Delta^C_{[K_v,K_e]}\left(G^{p^n}, G^{q^n}, \hat{f}^n, \dot{f}^n_{[K_v,K_e]}\right) = \left(C_{[K_v,K_e]}\left(G^{p^n}, G^{q^n}, \hat{f}^n_{[K_v,K_e]}\right) - C_{[K_v,K_e]}\left(G^{p^n}, G^{q^n}, \dot{f}^n_{[K_v,K_e]}\right)\right)^2 \tag{5.2}$$

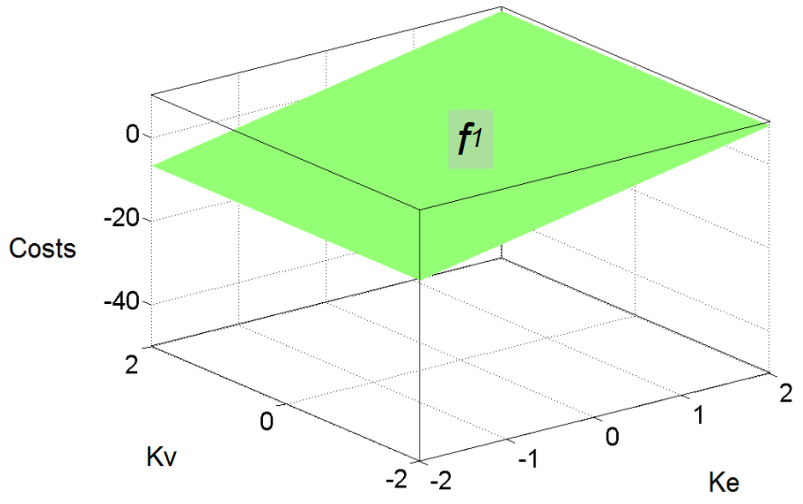


Figure 5.3. Edit Cost of f^1.

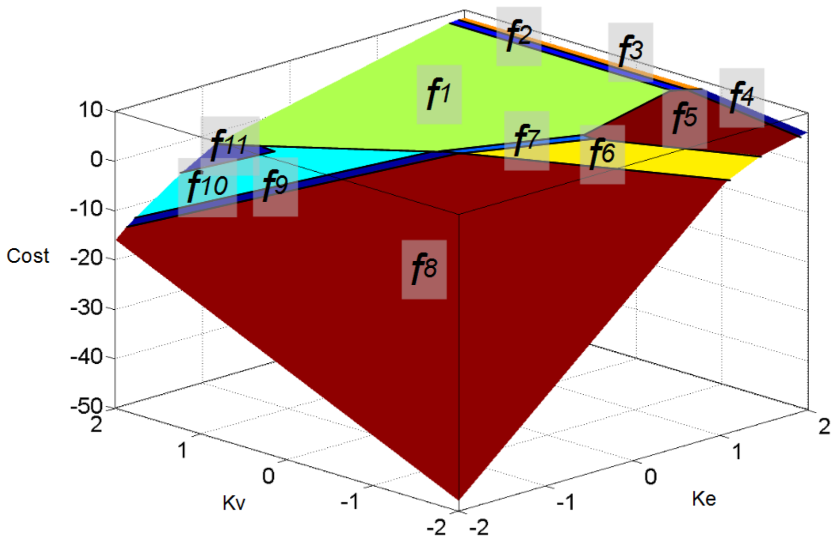


Figure 5.4. Edit Cost of the optimal correspondences.

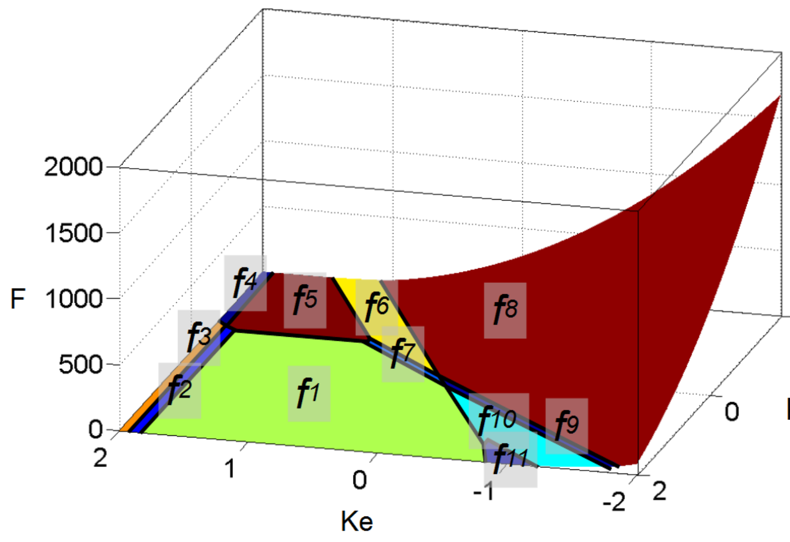


Figure 5.5. Edit Cost Error where the ground-truth correspondence is f^1.

Equation (5.3) shows the quadratic regularization term based on the inner product of the weights to be optimized in order to prevent overfitting and underfitting as in [44].

$$\Omega(K_v, K_e) = {K_v}^2 + {K_e}^2 \tag{5.3}$$

5.3 Experimental evaluation

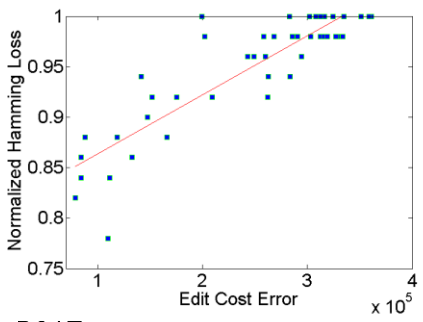
From the practical point of view, given a database, users wish to know which combination of parameters is the best such that algorithms obtain the best results (classification ratio, clustering quality, run time and so on). The aim of the proposed method is to automatically compute the insertion and deletion edit costs, such that the accuracy of the automatically obtained correspondence is maximised. Thus, these costs can be published together with the database with the information of witch algorithms have been used. We do not compare to any of the learning methods we commented (Table 1.1) because they do not directly learn the insertion and deletion edit costs. Papers [37, 38] learn the weights of a Self Organising Map and the parameters of some probability distributions such that the classification ratio is maximised. Papers [39, 40] learn the substitution weights considering nodes and edges have several attributes. Finally, paper [151] computes the insertion and deletion costs but it in limited costs rang and it needs a human to interact on the system at each iteration.

We performed experiments using the *BOAT*, the *EAST-PARK*, the *EAST-SOUTH* and the *RESIDENCE* sequences from the *Tarragona Rotation Zoom* repository and the *LETTER-HIGH* from the IAM repository, described in the Databases annex.

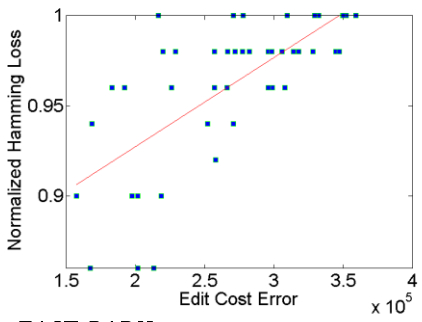
We used the *Fast Bipartite* [67] as the graph matching algorithm with different combinations of costs K_v and K_e. We defined the node substitution cost, C_{vs}, as the normalised Euclidean distance between node features, $C_{vs} \in [0,1]$ for *Tarragona Rotation Zoom* sequences. Recall that SURF features are composed of a vector of 64 elements. The structural dissimilarities costs C_{es} are computed using the *Degree* local sub-structure presented in Chapter 1. Moreover, the optimisation algorithm has been the *nelder-mead* method [152] with $\lambda = 1$.

5.3.1 Empirical validation of the Edit Cost Error

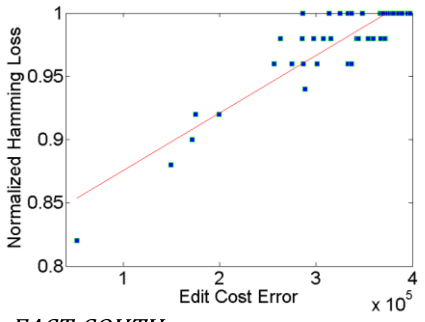
As in section 4.4.1 the aim of this section is to shown empirically the correlation between the *Edit Cost Error* and the *hamming distance* (normalised by the order of the graph). Figure 5.6 shows this correlation in all pairs of graphs in the databases, taken arbitrarily costs $K_v = 1$ and $K_e = 1$. We can deduce a positive linear relation although in some of the sequences there is a high linear approximation error. We show these scatters to empirically validate our initial assumption that when the *Edit Cost Error* increases, the *hamming distance* so it does.



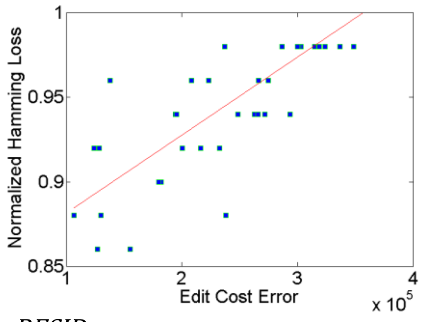
a. *BOAT*



b. *EAST-PARK*



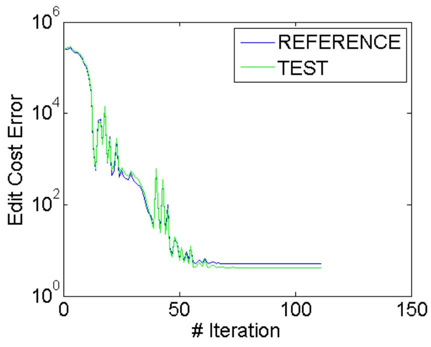
c. *EAST-SOUTH*



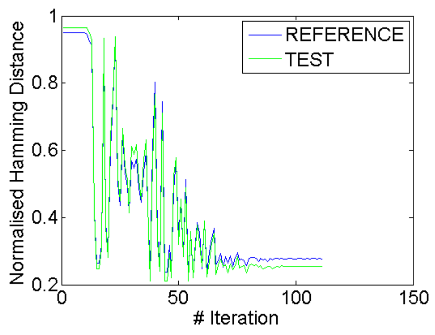
d. *RESID*

Figure 5.6. Hamming distance (normalised) versus Edit Cost Error.

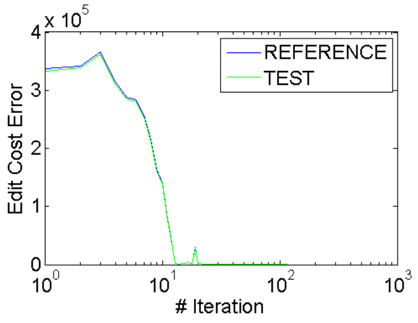
Figure 5.7 shows the evolution of the average *hamming distance* (normalised) and the average *Edit Cost Error* through the iterations of our learning algorithm for the *EAST-PARK* and *EAST-SOUTH* sequences. Initially, we set the costs $K_v = 1$ and $K_e = 1$. Once again, we realise when our learning algorithm forces the *Edit Cost Error* to the decrease, the *hamming distance* so it does.



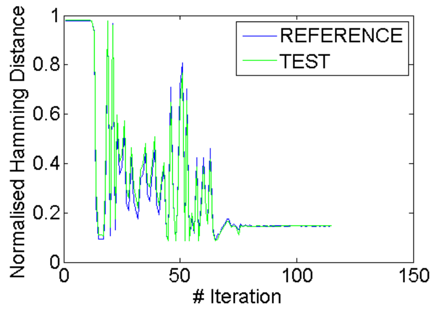
a. *BOAT*



b. *EAST-PARK*



c. *EAST-SOUTH*



d. *RESID*

Figure 5.7. Hamming distance versus Edit Cost Error.

5.3.2 Graph matching accuracy

The aim of this evaluation is to show the increase of accuracy when learned edit costs are used with respect to four different usual combinations of edit costs. The accuracy is defined as the average normalised number of nodes that have been correctly mapped: $1-\left(\sum_{n=1}^{N} d_H\left(\hat{f}^n,\dot{f}^n\right)\right)/(m\cdot N)$, m: number of nodes, N: number of graphs.

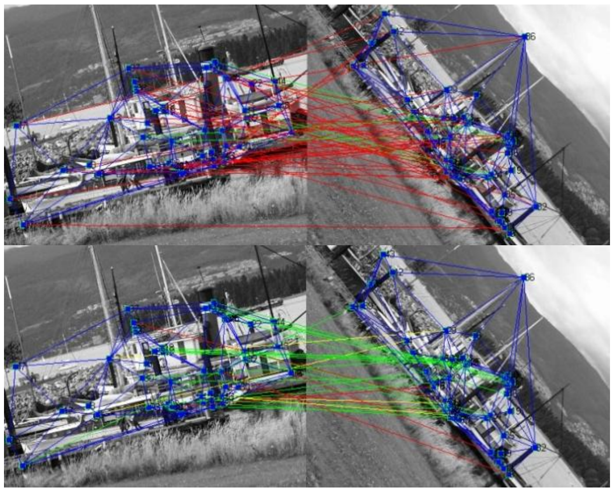
	Initial Costs			*Learned Costs*		
Databases	K_v	K_e	***Accuracy***	K_v	K_e	***Accuracy***
BOAT	1	1	0.0340	0.0487	-0.0021	0.7480
	0.5	0.5	0.0340	0.0487	-0.0021	0.7480
	1	0	0.0900	0.0487	-0.0021	0.7480
	0	1	0.0340	0.0487	-0.0021	0.7480
EAST-PARK	1	1	0.0260	0.0190	-0.0017	0.8160
	0.5	0.5	0.0280	0.0190	-0.0017	0.8160
	1	0	0.0680	0.0190	-0.0017	0.8160
	0	1	0.0260	0.0190	-0.0017	0.8160
EAST-SOUTH	1	1	0.0200	0.0135	0	0.8560
	0.5	0.5	0.0200	0.0135	0	0.8560
	1	0	0.0300	0.0135	0	0.8560
	0	1	0.0200	0.0135	0	0.8560
RESID	1	1	0.0440	0.0157	-0.0003	0.8360
	0.5	0.5	0.0440	0.0157	-0.0003	0.8320
	1	0	0.0820	0.0157	-0.0003	0.8320
	0	1	0.0440	0.0157	-0.0003	0.8320
ALL SEQUENCES	1	1	0.0310	0.0325	-0.0027	0.8095
	0.5	0.5	0.0315	0.0325	-0.0027	0.8095
	1	0	0.0675	0.0325	-0.0027	0.8095
	0	1	0.0310	0.0325	-0.0027	0.8095

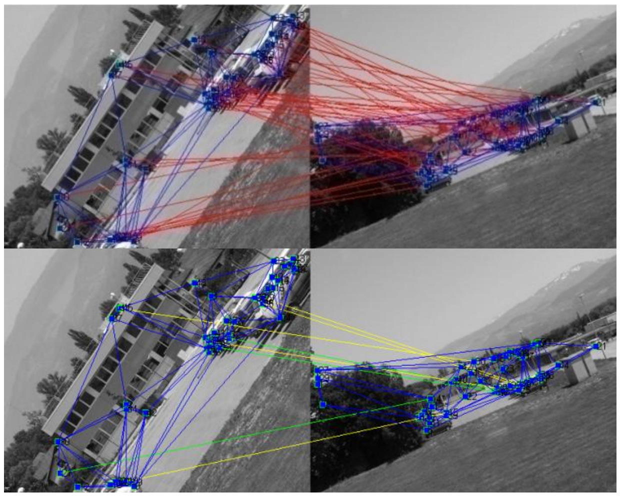
Table 5.1. Accuracy results and learned costs.

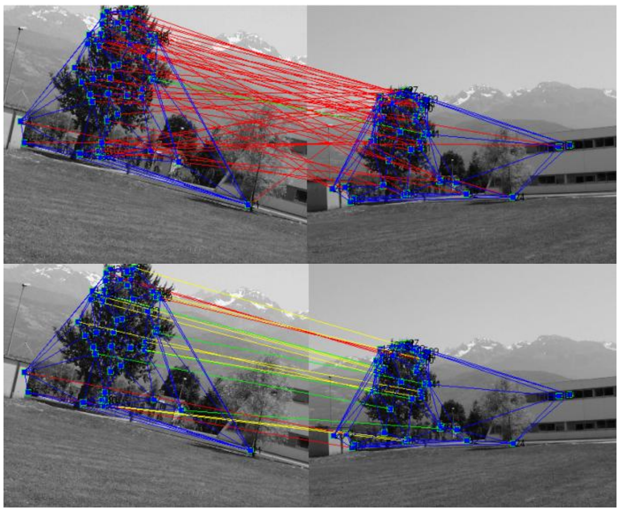
Table 5.1 shows the obtained average accuracies in the four sequences of *Tarragona Rotation Zoom* database. The first *Accuracy* column has been obtained using the initial costs (no learning process is performed). The second Accuracy column has been obtained using the learned costs. Note that, given a database, the learning algorithm converges at the same costs independently of the initial costs and the *Accuracy* drastically increases. The first two combinations of initial costs are the usual ones considering that

$C_{vs} \in [0,1]$ and $C_{es} \in \{0,1\}$. The last two combinations are considered to know if edges or nodes are useless. The last four rows are the results of learning all the graphs together. These results are used at section 5.3.3.

Figure 5.8 shows some examples of correspondences given the initial costs (up) and the learned costs (down). The incorrect mappings have drastically reduced (red lines) whereas the correct mappings (green lines) have increased. Note missing mappings (yellow lines) have also increased. This is because the structure (*Delaunay* triangulation) is not correct and the algorithm decides not map the nodes instead of wrongly mapping them.


a. *BOAT*


b. *EAST-SOUTH*


c. *EAST-PARK*

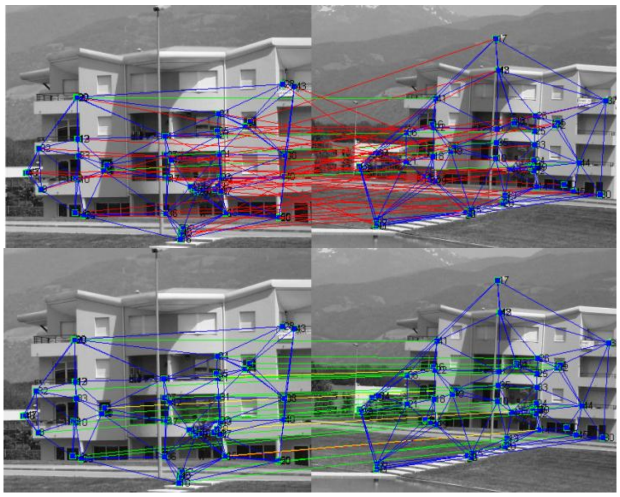
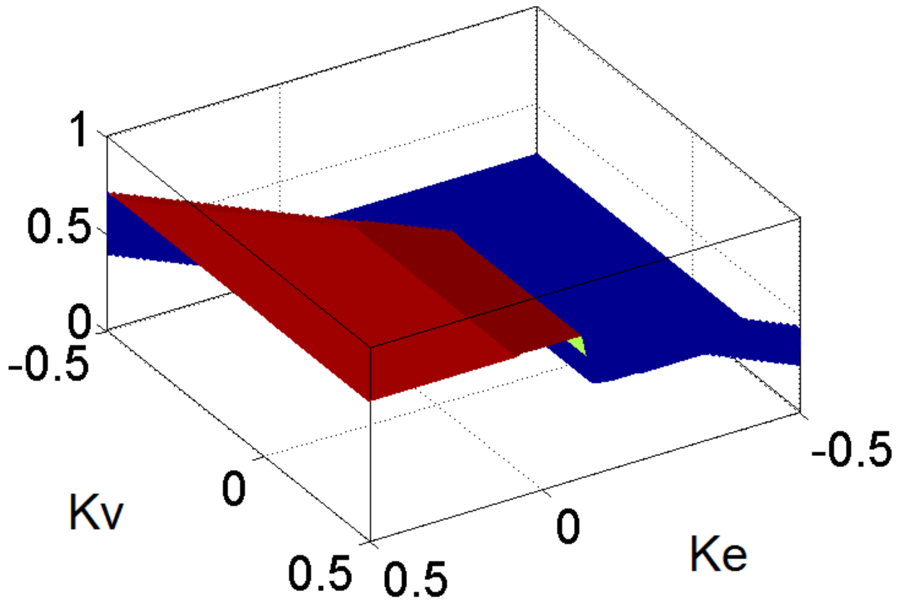

d. *RESID*

Figure 5.8. Some examples of correspondences. Up: $K_v = 1$ and $K_e = 1$. Down: Learned costs. Yellow lines: missing mappings. Green lines: correct mappings. Red lines: incorrect mappings.

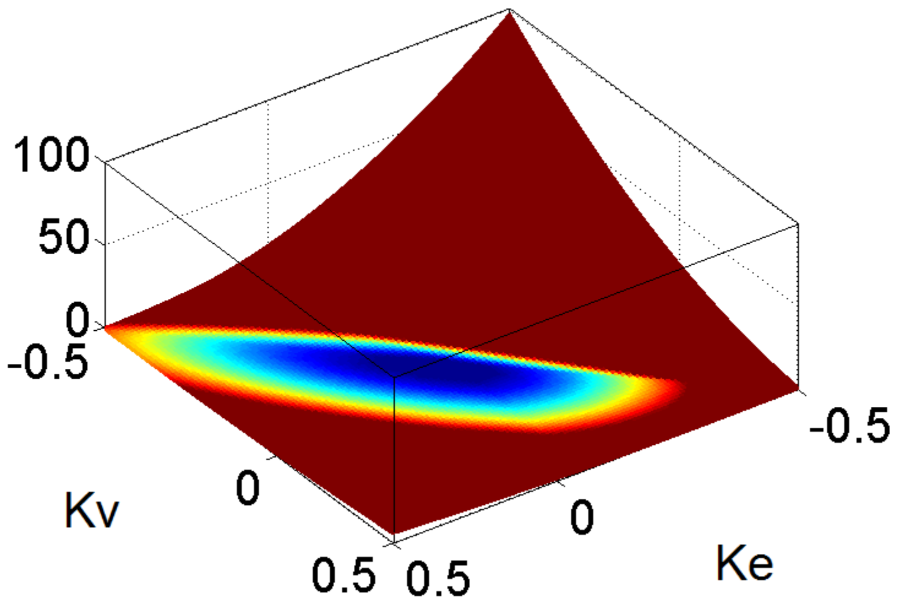
In the same way than Table 5.1, Table 5.2 shows the *Accuracy* given different combinations of initial costs applied to the *LETTER-HIGH* database. We observe that in all initialisation values, the learning algorithm converges to a similar edit costs. For this reason, we computed the *Accuracy* and the *Edit Cost Error* (equation (5.2)) given all possible combinations of costs from -0.5 to 0.5 with a resolution of 0.01 (in this case, there is no a learning process). Figure 5.9 shows the obtained values. We observe that in this database, the *Edit Cost Error* defines a concave shape and also its global minimum is located at the point where the *Accuracy* obtains the global maximum. Therefore, we have empirically validated that while minimising the *Edit Cost Error*, the *Accuracy* is maximised.

	Initial Costs			***Learned Costs***		
Databases	K_v	K_e	***Accuracy***	K_v	K_e	***Accuracy***
LETTER-HIGH	0.5	0.5	0.6680	0.0198	0.0072	0.7197
	-0.5	-0.5	0.2186	0.0198	0.0072	0.7197
	0.5	-0.5	0.2186	0.0198	0.0072	0.7197
	-0.5	0.5	0.3756	0.0198	0.0072	0.7197
	0.5	0	0.5688	0.0198	0.0072	0.7197
	-0.5	0	0.2186	0.0198	0.0072	0.7197
	0	0.5	0.6711	0.0198	0.0072	0.7197
	0	-0.5	0.2186	0.0198	0.0072	0.7197

Table 5.2. Accuracy results and learned costs.



a. *Accuracy*



b. *Edit Cost Error*

Figure 5.9. Accuracy and Edit Cost Error in LETTER-HIGH experiments.

5.3.3 Classification and Clustering Accuracy

Figure 5.10 shows the confusion matrices of the four sequences of *Tarragona Rotation Zoom* when the distance is computed through costs $K_v = 1$ and $K_e = 1$ and through the learned costs ("*ALL SEQUENCES*" in Table 5.1). In the second case, the whole graphs are properly classified.

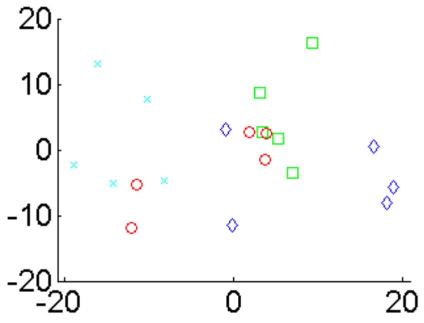
	BOAT	EAST PARK	EAST SOUTH	RESID
BOAT	0.6	0.4	0	0
EAST PARK	0.2	0.4	0	0.4
EAST SOUTH	0	0.2	0.6	0.2
RESID	0	0	0	1

a. $K_v = 1$ *and* $K_e = 1$

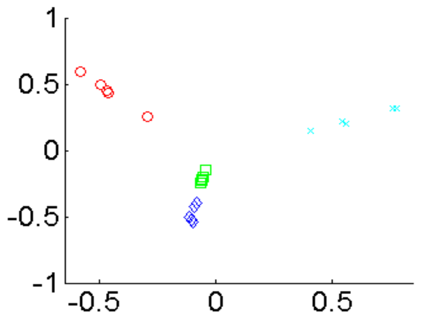
	BOAT	EAST PARK	EAST SOUTH	RESID
BOAT	1	0	0	0
EAST PARK	0	1	0	0
EAST SOUTH	0	0	1	0
RESID	0	0	0	1

b. $K_v = 0.0325$ *and* $K_e = -0.0027$

Figure 5.10. Confusion matrices.



a. $K_v = 1$ *and* $K_e = 1$



b. $K_v = 0.0325$ *and* $K_e = -0.0027$

Figure 5.11. Multidimensionality reduction of the distance space. □ BOAT, ○ EAST PARK, ◇ EAST SOUTH, ✕ RESID.

Figure 5.11 shows the result of applying multidimensionality reduction to the space of distances between *attributed graphs*. Each symbol represents a different sequence. On the left, costs are $K_v = 1$ and $K_e = 1$. On the right, costs are the mean of the learned costs in the four sequences (Table 5.1). Clearly, graphs are grouped when learned costs are used to compute the Edit distance between graphs making the unsupervised clustering process to be really easy.

Chapter 6 Graph Databases Based on Metric-Trees of Graph-Class Prototypes

Metric-trees are well-know structures used to speed-up queries in databases. In this chapter, we evaluate the applicability of metric-trees to graph databases. In classical schemes based on metric-trees, the routing information kept in a metric-tree, the node is a selected element from the sub-cluster that represents. Nevertheless, defining a graph that represents a set of graphs is not a trivial task. In this chapter are evaluated the representational power of different *Graph-Class Prototypes* as routing nodes of the metric-tree. The considered prototypes are: *Median Graphs*, *Set Median Graphs*, *Closure Graphs*, *First-Order Random Graphs*, *Function-Described Graphs* and *Second-Order Random Graphs*.

6.1 Graph-Class Prototypes

This section presents the most important models used to represent a set of *attributed graphs*. The first three models are non-probabilistic and the last three models are probabilistic. In each model, we briefly comment how to obtain the prototype given a computed *Common Labelling* and a set of *attributed graphs*.

The *Common Labelling* is a function that assigns all the graph nodes of a set of *attributed graphs* to a *virtual structure* that represents the different nodes. That is, nodes in the *attributed graphs* that represent the same local part of the object or image have to be represented by only one node in the virtual structure.

6.1.1 Median Graph

A *Median Graph* [59, 58, 61, 62], is an *attributed graph* that minimizes the sum of distances between it and all the graphs in the set of *attributed graphs*. It has exactly the same information and domain than the *attributed graphs* that it represents. Notice that it is usually not a member of the set, and in general, more than one *Median Graph* may exist for a given set of graphs. Formally, *Median Graph*. Given a set of graphs $\Gamma = \{G^1, G^2, \dots, G^N\}$ in a set U and a distance between *attributed graphs* $d(G^p, G^q)$, the *Median Graph* is defined as follows.

$$\bar{g} = \underset{g \in \mathrm{U}}{argmin} \sum_{G^i \in \Gamma} d\left(g, G^i\right) \tag{6.1}$$

That is, the Median Graph is the graph $g \in \mathrm{U}$ that minimizes the sum of distances to all graphs in Γ. The set U represents the universe of *attributed graphs* in a specific domain.

The computation of a *Median Graph* is a *NP-complete* problem. Nevertheless, several suboptimal methods to obtain approximate solutions for the *Median Graph*, in reasonable time. These methods apply some heuristic functions in order to reduce the complexity of the graph distance computation and the size of the search space.

An alternative to *Median Graphs*, which is less computationally demanding, is the *Set Median Graph*. The difference between the two models consists in the search space where the *Median Graph* is looked for. The search space for the *Median Graph* is the whole universe of *attributed graphs* U.

6.1.2 Set Median Graph

In contrast, the search space for the *Set Median Graph* [54], is simply the set of graphs that represents Γ. Formally, *Set Median Graph*. Given a set of graphs $\Gamma = \{G^1, G^2, \ldots, G^N\}$ and a distance between *attributed graphs* $d(G^p, G^q)$, the *Set Median Graph* is defined as follows.

$$\bar{g} = \underset{g \in \Gamma}{\operatorname{argmin}} \sum_{G^i \in \Gamma} d\left(g, G^i\right) \tag{6.2}$$

The computation of the *Set Median Graph* is exponential in the size of the graphs, due to the complexity of graph edit distance, but quadratic with respect to the number of graphs in the set. Nevertheless, *Set Median Graphs* have less ability to capture the information of the cluster than *Median Graphs* due to the reduction of the search space.

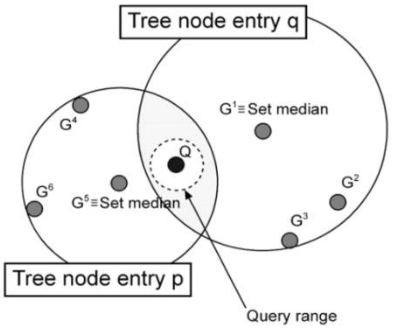


Figure 6.1. Clusters represented by a Set Median Graph.

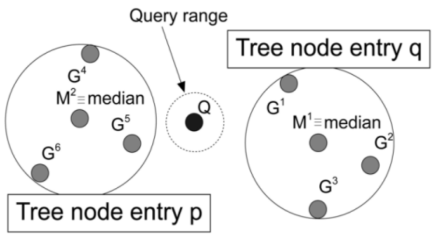


Figure 6.2. Clusters represented by a Median Graph.

The advantages of using *Median Graphs* as routing elements in a metric-tree are manifold. The main effect of using them is the reduction of the overlap between sub-clusters, due to the radius of the covering region, d^h, can be more tightly adjusted. In fact, if we use the *Median Graphs* as a routing element, the radius of the covering region has to be equal or lower than the radius of the covering region represented by a *Set Median Graph* of the cluster.

Figure 6.1 and Figure 6.2 shows an example of representing the cluster by a *Set Median Graphs* (Figure 6.1) and by a Median Graph (Figure 6.2). Both figures show the same 6 elements in two sub-clusters and the radius of their covering regions. Suppose a hypothetical query graph Q with a query range represented by the outer doted circle. The execution of the search behaves very different on both representations. In the *Set Median Graph* approach, neither entry *p* or *q* holds for equations (6.1) and (6.2), so the sub^q and sub^p must be explored. However, in the *Median Graph*, (6.2) holds for both tree node entries *p* and *q*. Consequently, it can be assumed that none of the entries contain any desired graph. Thus, they can be discarded and not explored.

6.1.3 Closure Graph

A *Closure Graph* [55], is another model graph that the structure of the attributes is different to the *attributed graphs* that represent. The structurally similar nodes of the set of *attributed graphs* that have different attribute values are represented in the *Closure Graph* with only one node but with more than one attribute. *Closure Graphs* need the attributes at nodes of the graphs to be discrete, if it is not the case; an extra discretization process has to be performed. *Closure Graphs* needs few more physical space than the *Median Graphs*.

Formally, a *Closure Graph* $M = \left(\widehat{\Sigma_v}, \widehat{\Sigma_e}, \widehat{\gamma_v}, \widehat{\gamma_e}\right)$ is a graph where the node and edge attribute domains are a set of domains of the nodes and edges of the *Attribute Graphs* *{Δ_v, Δ_v, ..., Δ_v}* and *{Δ_e, Δ_e, ..., Δ_e}*. The specific value are *{a_1, a_2, ..., a_t}* and *{b_1, b_2, ..., b_h}*. We synthesise a *Closure Graph* from a set of *attributed graphs* Γ and a *Common Labelling* ψ as follows:

$$\widehat{\gamma_v}\left(\widehat{v_j}\right) = \left\{\cup\, a_k \,\middle|\, \exists a_k = \gamma_v\left(h^{i^{-1}}\left(\widehat{v_j}\right)\right), 1 \le i \le |\Gamma|, \gamma_v\left(h^{i^{-1}}\left(\widehat{v_j}\right)\right) \neq \phi\right\}$$
$$\widehat{\gamma_e}\left(\widehat{e_{qj}}\right) = \{\cup\, b_k | \exists b_k = \varepsilon, \varepsilon \neq \phi\} \qquad \textbf{(6.3)}$$

Being ε the attribute value of the edges $e_{h^{i^{-1}}(\widehat{v_q}), h^{i^{-1}}(\widehat{v_j})}$ in G^i. The main idea of (6.3) is that nodes or edges of the *Closure Graph* can take all values that nodes or edges of the *attributed graphs* have taken. In the case that there does not exist a node or edge in the *attributed graph*, its value is not considered.

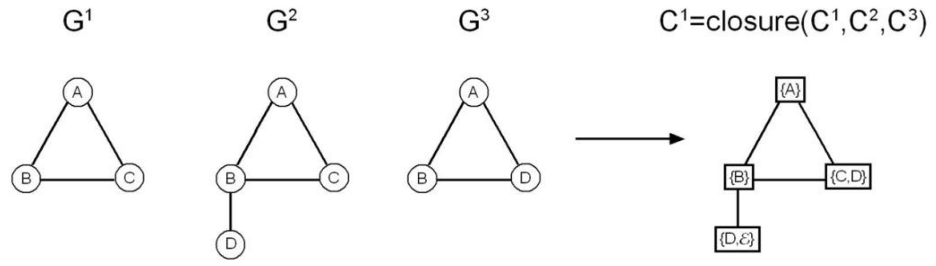


Figure 6.3. Example of a Closure obtained by 3 attributed graphs.

Practical evaluations show that *Median Graphs* and *Closure Graphs* tend to generalize to much the set that they represent; allowing graphs that are distant from the ones that have not been used to synthesize them. To alleviate this weakness, the following probabilistic models have been defined.

6.1.4 First-Order Random Graph (FORG)

A *First-Order Random Graph* (FORG) [22], is a model graph that contains first-order probabilities on nodes and edges to describe a set of *attributed graphs*. To deal with the first-order probabilities, there is a random variable associated with each node or edge, which represents the attribute information of the corresponding graph nodes and edges in the set of *attributed graphs*. This random variable has a one-dimensional probability density function defined over the same attribute domain of the *attributed graphs*, including a null value that denotes the non-instantiation of a FORG graph node or edge in an *attributed graphs*.

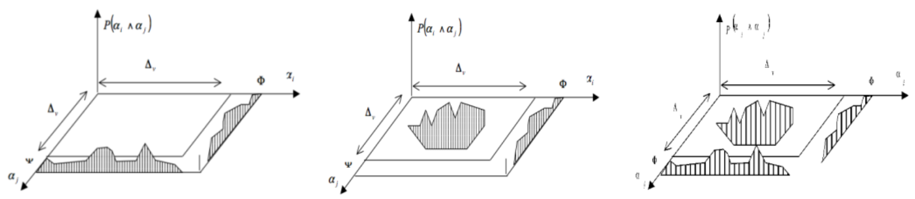
This model was the first probabilistic one that appeared in the literature to represent a set of *attributed graphs*. It assumes that the *attributed graphs* in a set or cluster had similar local parts. Nevertheless, in practical applications, some graphs can be quite different despite of belonging to the same class. For this reason, representing a set of graphs with only first order probabilities seems to be too restrictive.

6.1.5 Function-Described Graph (FDG)

A *Function-Described Graph* (FDG) [23, 66], is a model graph that appeared with the aim of overcoming the representational power of FORGs. It contains first-order probabilities of attributes and second-order structural information to describe a set of *attributed graphs*. The first order information was represented in the same way than FORGs trough probability density functions. The second-order structural information is qualitative information of the second-order joint probability of each pair of nodes or edges. This information is represented by binary relations called *Antagonisms*, *Occurrences* and *Existences* between nodes and edges. FDGs increased the representational power at the cost of increasing also the required physical space.

Two nodes or edges are *antagonistic* if they have never taken place together in any graph used to synthesise the FDG although these two nodes or edges are included in the FDG as different elementary parts. There is an *occurrence* relation between two nodes or edges of the FDG if always that one of related nodes or edges in the graph has appeared; also the other node or edge of the same graph has appeared. Finally, there is an *existence* relation between two nodes or edges if all the graphs in the class described by the FDG have at least one of the two nodes or edges.

6.1.6 Second-Order Random Graph (SORG)



a. *Antagonisms* b. *Occurrences* c. *Existences*

Figure 6.4. Histograms of a Second-Order Random Graph.

A *Second-Order Random Graph* (SORG) [24], is a probabilistic model closely related to FDGs. The main difference lies in the fact that the second-order structural information is not defined as binary relations but with the specific information of the second-order joint probability. Thus, the physical space needed to represent SORGs is much higher than FDGs but also its ability to represent the set of *attributed graphs* increases. In [23, 66], it is shown how to convert a SORG to an FDGs simply by analysing the second-order probabilities and deciding if the binary relations hold.

6.2 Synthesis of Graph-Class Prototypes

Two types of methods exist to generate *Graph-Class Prototypes* from a given set of graphs [23, 24]. We assume the structure of the metric-tree has been computed (section 6.3) and we have to compute the *Graph-Class Prototype* and the radius of the cluster d^M. The first method is based on a *Hierarchical Synthesis*. The second one is based on a *Global Synthesis* based on a *Common Labelling* [155, 156, 157].

6.2.1 Hierarchical Synthesis of Median Graphs

In the *Hierarchical* method, each *Graph-Class Prototype* is computed only using two *Graph-Class Prototypes* or *attributed graphs* at a time. Therefore, a *Common Labelling* is not needed and *Graph-Class Prototypes* are computed as pairwise consecutive computations of other *Graph-Class Prototypes* obtained in lower levels of the tree.

6.2.2 Global Synthesis of Median Graphs using a Common Labelling

In the *Global Synthesis*, each *Graph-Class Prototype* is computed using the whole set of *attributed graphs* in the cluster that the metric-tree node represents, independently of whether the metric-tree node has other nodes as descendants in the tree. The first step of this method computes a *Common Labelling* from the *attributed graphs* of the sub-cluster and the second step obtains the *Graph-Class Prototype*. We have used two different *Common Labelling* algorithms, which have the main feature that are independent of the *Graph-Class Prototype* to be synthesised. The first one is based on the *Graduated Assignment* [65] and the second one is based on a *genetic algorithm* [64].

Note that the *Set Median* is a special prototype since it does not need to be synthesised. The *Set Median* is the graphs of the cluster that has the minimum distance between it and the other graphs.

6.3 Metric-trees

A metric-tree [57] is a scheme to partition a database in a hierarchical set of clusters, collecting similar objects. Each cluster has a routing object and a radius providing an upper bound for the maximum distance between the reference object and any other object in the cluster. Triangle inequality can be used during the access to the database to prune clusters that are bound out of an assigned range from the query. Figure 6.5 shows a graphical representation in a bi-dimensional space of two graphs (G^{11} and G^{12}) together with the routing element M^4 that represents the cluster that they belong to. The circumference represents the area influenced by the cluster of

M^4 with its radius. The dashed partial circumference represents another cluster that contains *M^4* cluster and other ones.

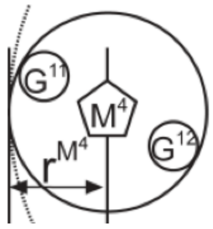


Figure 6.5. Graphical representation of a cluster.

Formally, a metric-tree is a tree of nodes. Each node contains a fixed maximum number of *m* entries, *<node> := {<entry>}^{m}*. In turn, each entry is constituted by a routing element *M*; a reference to the father *r^M* of a sub-index containing the element in the so-called covering region of *M*; and a radius *d^M* providing an upper bound for the distance between *M* and any element in its covering region, *<entry> := {M, r^M, d^M}*. During retrieval of an element *Q*, triangular inequality is used to support efficient processing of queries. To this end, the distance between *Q* and any element in the covering region of a routing element *M* can be max-bounded using the radius *d^M* plus the distance between *Q* and *M*.

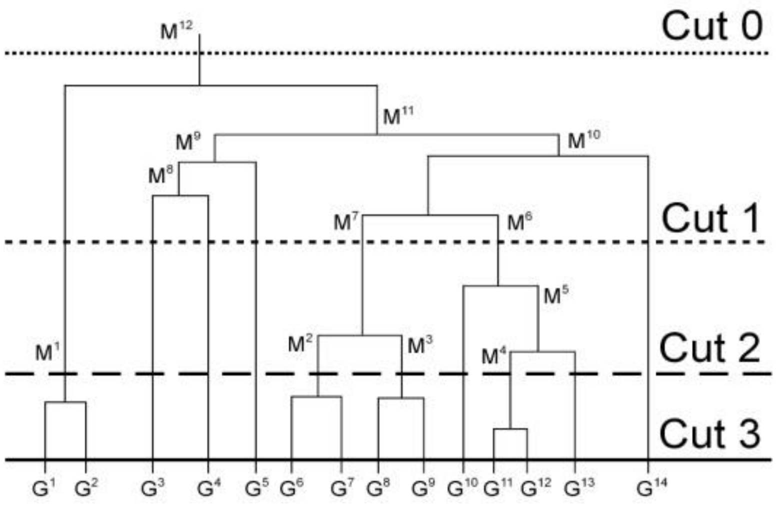


Figure 6.6. Example of a dendogram.

The metric-tree can be constructed using different schemes for the insertion of a new element and the selection of the routing element [57]. In this thesis, we use a general construction methodology from which we are able to construct a metric-tree independently of the type of the routing element. We use a non-balanced tree constructed through a *hierarchical clustering* algorithm and complete *linkage clustering* [154]. In this way, given a set of graphs, the distance matrix over the whole set is computed and then a dendogram is constructed. Using the dendogram and some horizontal cuts, a set of partitions that clusters the graphs in the database is obtained. With these partitions the metric-tree is generated. Finally, the information on the routing elements in the metric-tree is inserted, M and d^M.

In our case, M is a *Graph-Class Prototype* and d^M is the maximum distance between the *Graph-Class Prototype* and any of the graphs in the covering region. Figure 6.6 shows an example of a dendogram. Graphs that represent components G^i are placed on the leaves of the dendogram and the routing elements M^j are placed on the junctions between the cuts and the horizontal lines of the dendograms.

Dendogram of Figure 6.6 defines 4 different partitions. Figure 6.7 shows the obtained metric-tree. Note that in some tree nodes, there are Class-*Graph Prototypes* (M^j) together with original graphs (G^i).

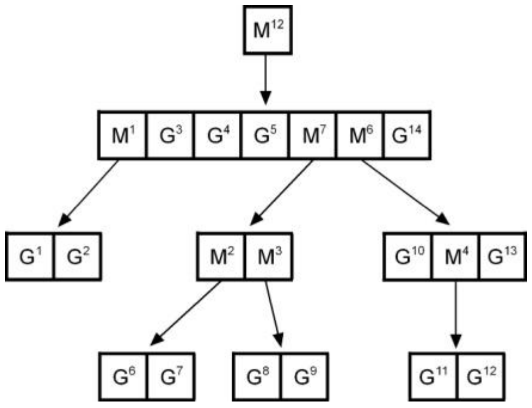


Figure 6.7. The obtained metric-tree.

Two different types of queries can be performed to databases organised by metric-trees: KNN queries and *similarity* queries [63].

6.4 Queries on Graph Databases

Two different types of queries can be performed to databases organised by metric-trees: KNN queries and *similarity* queries [63].

6.4.1 Similarity queries on Graph Databases

To perform *similarity* queries in metric trees, the tree is analyzed in a top down fashion. Specifically, if d_{max} is the range of the query and Q is the query graph, the following conditions are employed, at each node of the tree, to check whether all the elements in the covering region of *M*, sub^M, can be discarded, accepted or need more exploration. The conditions are based on the evaluation of the distance between the routing element and the query element *d(Q,M)*.

If the following condition holds, we reject all elements deeper from the routing element.

$$d(Q,M) \geq d_{max} + d^M \rightarrow \textit{No element in } sub^M \textit{ is acceptable} \qquad \textbf{(6.1)}$$

In a similar manner, the following condition checks whether all the elements in the covering region of *M*, sub^M, fall within the range of the query. In this case, all the elements in the region can be accepted:

$$d(Q,M) \leq d_{max} + d^M \rightarrow \textit{Every element in } sub^M \textit{ is acceptable} \qquad \textbf{(6.2)}$$

In the critical case that neither of the two inequalities holds, the covering region of *M*, sub^M, may contain both acceptable and no acceptable elements, and the search must be repeated on the sub index sub^M. In the case where knowledge in the tree nodes is not the *Median Graph* but a *Graph-Class Prototype* d_{max} needs to be adapted to each *Graph-Class Prototype* defining a conversion factor as a parameter or a function.

6.4.2 Nearest Neighbours on Graph Databases

The most common method to perform Nearest-Neighbour queries uses a branch-and-bound algorithm, which utilises two global structures: A priory queue PR that stores the possibly fruitful tree nodes to be explored and an array NN that stores the best K elements found until the moment. PR is a queue of pointers to active sub-trees, i.e. sub-trees where objects that may be near to the query element can be found. The K values of NN are initialised to a null element. PR is initialised with one element which is the root of the metric tree. Note that PR does not have a maximum number of elements.

Let Q be a query element and $d_{max}^{NN}(Q)$ the maximum distance from Q to any element in NN. The distance $d_{max}^{NN}(Q)$ is initialised to infinitive. In each iteration of the search algorithm, the tree node in PR with lower distance to Q is selected, let this node be named N and its children be N^1, … , N^t. The distances between Q and all the children of N must be computed.

If son N^i is a *routing node*, this node is inserted in PR if $d(N^i, Q) - d^{N^i} \leq d_{max}^{NN}(Q)$. This insertion is done due to it is possible to find an element with lower distance than the ones already found. On the contrary, it is not possible that any of their descendants have a distance lower than $d_{max}^{NN}(Q)$ and so, none of the nodes and leaves of the branch are explored.

If son N^i is a *leaf*, that is, a database element, and $d(N^i, Q) \leq d_{max}^{NN}(Q)$, array NN and $d_{max}^{NN}(Q)$ are updated to consider this element in the following way. There are two possibilities:

If all NN has its K positions full with leaves, then the element with higher distance is discarded. Moreover, the distance $d_{max}^{NN}(Q)$ is updated to be the maximum distance from Q to any element in NN. This new value has to be lower than the previous value since $d_{max}^{NN}(Q)$ acts as a dynamic maximum search distance of range queries.

If NN does not have its K positions full with leaves, then, the new tree leave N^i (that is, an element of the database) is inserted in an empty position of NN. Related to the distance $d_{max}^{NN}(Q)$, there are again two cases. If NN continues to be non full, then, the distance keeps with its initial value, which is infinitive. But, if the new situation of NN is that all the elements are used, $d_{max}^{NN}(Q)$ takes the maximum value of the distances between the leaves in NN and Q.

The routing element M can be defined as one of the elements of the sub-cluster or a new element that represents the elements of the sub-cluster. The main effect of using a new prototype might be the reduction of the overlap between sub-clusters, due to the radius of the covering region can be more tightly adjusted.

6.5 Experimental Evaluation

We have selected four datasets form the repository presented in [66], described in the Databases annex. The datasets are *COIL*, *LETTER (HIGH* and *LOW*) and *GREC*.

6.5.1 Graph prototypes for classification

Table 6.1 presents classification accuracies achieved with different *structural* [59, 22, 66, 24, 55] and *embedding* [58, 158, 159] methods. The embedding methods in [158, 159] are based on representing each graph with a vector of distances which related each graph to a set of prototypes extracted from the training set. Different prototype selection methods are presented: *sps-c*, *bps-c* and *k-cps-c*.

Once selected the embedding prototypes are performed, training is done using a SVM. In the original article, these methods are compared with the KNN classifier under the *Graph Edit Distance*. The method in [58] is addressed to compute the *Median Graph* to later apply a KNN classifier. The *Median Graph* is computed in an iterative form using an embedding space, the selected embedding is inspired in [158, 159]. It is important to highlight that, results extracted from [158, 159] may not be evaluated with the same version of the dataset presented here, since, even the dataset we used is downloaded from the same website cited in [158, 159], the number of training, validation and test elements do not seem to correspond. Even though, we assume that results can be compared since the results we obtained with the reference KNN method seem to correlate with the results obtained in [158, 159]. Under these considerations, obtained results show that structural/classical methods achieve recognition ratio on the same range as new methodologies based on graph embeddings. Some structural methods, such as the *Generalized Median Graph*, obtain less classification ratio than the embedding methodologies.

However, more advanced structural methods, such as *First Order Random Graphs*, *Function Described Graphs*, *Second Order Random Graphs*, obtain greater recognition ratio, even if the classification algorithm, *i.e.* the KNN, is much simpler than the SVM used in [158, 159]. With respect to the embedding method in [58], we see that the improvement of structural methods is notable. More advanced classification algorithms may further increase the recognition ratio. With respect to the approach with different representatives per class on structural algorithms, it is possible to note a general tendency on the decrease of the recognition ratio. This tendency increases in *Closure Graphs*. Thus, we could conclude that, in these datasets, it is better to use all the information available to generate a single prototype. One could think that some clustering schema to group elements, instead of random selection, should improve results. However, since, in this particular dataset, noise on elements is uniformly distributed, a random selection algorithm is as valid as any other grouping scheme.

Alg. type	**Method**	**LETTER LOW**	**LETTER HIGH**	**GREC**	**Average**
Ref. systems	KNN (T1)	0.91	0.62	0.86	0.80
	KNN (T2)	0.89	-	-	-
	KNN (T3)	0.91	0.81	0.96	0.89
	KNN (T4)	0.94	0.82	0.95	0.90
	KNN (T5)	-	-	0.98	-
Embedding method	Emb. Kernel (T1)	0.92	0.74	0.89	0.85
	sps-c (T2)	0.92	-	-	-
	bps-c (T2)	0.92	-	-	-
	k-cps-c (T2)	0.93	-	-	-
	Set Median (T5)	-	-	0.77	-
	Median Graph (T5)	-		0.79	-
Structural methods	Set Median	**0.96**	0.69	0.85	0,83
	Median Graph	0.89	0.70	**0.91**	0.83
	Closure Tree	0.69	0.22	0.56	0.49
	FORG	0.92	**0.79**	0.83	0.84
	FDG	0.92	**0.81**	0.83	**0,85**
	SORG	**0.93**	**0.79**	**0.89**	**0.87**

Table 6.1. Comparative study between embedding and structural methods for graph classification. Tags indicate: (T1) results extracted from [158], (T2) results extracted from [159], (T3) distance computed with the Graduated Assignment [32], (T4) distance computed with the Bipartite [25] (T5) results extracted from [58].

6.5.2 Metric-trees queries

We aim to compare four metric-trees that have been synthesised with different types of routing information. In each test, only one metric-tree is constructed with some graphs of the reference set.

The number of graphs used to construct the metric-trees is 50 randomly extracted from the reference set, with 7 dendogram cuts (number of levels of the metric-tree, due to the computational runtime required by the method used to generate the metric-tree). The distances to set the cuts are the following; distance of cut 0 = D_{max}, distance of cut 1 = $D_{max}\cdot 6/7$, distance cut 2 = $D_{max}\cdot 5/7$, ..., distance of cut 6 = $D_{max}/7$. Where D_{max} is the maximum distance of any two graphs of the metric-tree. Each experiment consists of 50 graph queries over the metric-tree. For the KNN queries, the number of elements to be retrieved are 3 (k = 3) and for the D-max queries, the range of the query is $0.6\cdot D_{max}$ ($d_{max} = 0.6\cdot D_{max}$).

6.5.2.1 Quality indices

Four quality indices have been used: *Access ratio*, *precision*, *recall* and *F-measure*.

6.5.2.1.1 Access Ratio

This index evaluates the capacity of the metric-tree to properly route the queries. Given a query graph Q, this index is the number of accessed nodes and leaves of the metric-tree or the number of matching performed, R. Finally, it is normalized by the number of graphs used to generate the metric-tree, N. If the *access ratio* is higher than 1, then it is faster not to use a metric-tree since without the metric-tree, the system would perform less comparisons.

$$AccessRatio(Q, K, T) = \frac{R}{N} \tag{6.4}$$

6.5.2.1.2 Precision

Precision is the number of items correctly labelled as belonging to the class divided by the total number of elements labelled as belonging to the positive class for KNN and under acceptable distance for DMAX. We obtain this metrics as follows, from the K graphs returned by the metric-tree with minimum distance between them and Q for DMAX, and the number of elements of the same class as Q for KNN, and the number of true positives *Tp*.

$$Precision(Q,K,T) = \frac{T_p}{K} \tag{6.5}$$

6.5.2.1.3 Recall

Recall is the number of items correctly labelled as belonging to the class divided by the total number of elements that actually belong to the class for KNN and under acceptable distance for DMAX. We obtain this metrics as follows, from the C graphs returned by all the reference set with minimum distance between them and Q for DMAX, and the number of elements of the same class as Q for KNN, and the number of true positives *Tp*.

$$Recall(Q,K,T) = \frac{T_p}{C} \tag{6.6}$$

6.5.2.1.4 F-Measure

The *F-measure* is a measure of a test's accuracy computed through the *precision* and *recall*.

$$F\text{–}mesure = \frac{2\, Precision\, Recall}{Precision + Recall} = \frac{2\, Tp}{C + K} \tag{6.7}$$

6.5.2.2 Results

Table 6.2 and Table 6.3 show the *access ratio* in nearest neighbour and *similarity* queries. Lower is the *access ratio* faster is the query. Besides, if the *access ratio* is greater than 1, the number of comparisons done using the metric-tree is higher than if there was no metric-tree and the graphs of the whole database where all compared. This situation does not appear in the KNN queries, which means that it is worth to structure the database in a metric-tree. On the contrary, some values of Table 6.3 (*similarity* query) are greater than 1. To reduce this problem, d_{max} whole have to be reduced but we preferred to use the same value for all the experiments for the compactness of the result values. Besides, some cells of Table 6.3 have value 0.02. This is because, given a query, only the root node of the metric-tree is explored. Considering that the metric-tree has been built using 50 graphs, the value comes from $0.02 = 1/50$. Again, this problem could be solved by adapting d_{max} value to each *Graph-Class Prototype*. In general, *Median Graphs* are the prototypes with better *access ratio*. So, they obtain faster queries (except for the commented extreme values, 0.02).

Access Ratio (k=3)	**Synthesis**	**COIL**	**LETTER-LOW**	**LETTER-HIGH**	**GREC**
Set Median	-	0.54	0.34	0.35	0.45
Generalise Median	Hierarchical	**0.38**	**0.31**	**0.33**	**0.37**
Generalise Median	Genetic	0.40	0.38	0.36	0.57
Generalise Median	Graduated Assignment	0.44	0.34	0.37	0.40
Closure Graph		0.65	0.33	0.57	0.80
FORG		0.42	0.35	0.42	0.47
SORG		0.35	0.33	0.36	0.38
FDG		0.45	0.36	0.45	0.56

Table 6.2. Access ratio on nearest neighbour queries.

Access Ratio ($d_{max} = 0.6 \cdot D_{max}$)	**Synthesis**	**COIL**	**LETTER-LOW**	**LETTER-HIGH**	**GREC**
Set Median	-	**0.02**	1.03	1.46	0.06
Generalise Median	Hierarchical	0.47	0.99	1.34	1.07
Generalise Median	Genetic	0.92	1.65	1.66	0.78
Generalise Median	Graduated Assignment	0.89	0.98	1.30	0.94
Closure Graph		**0.02**	**0.02**	**0.02**	**0.02**
FORG		**0.02**	**0.02**	**0.02**	**0.02**
SORG		**0.02**	2.27	2.78	**0.02**
FDG		**0.02**	0.47	0.04	**0.02**

Table 6.3. Access ratio on similarity queries.

Table 6.4 and Table 6.5 show the mean *precision*. SORGs and *Closures* are the prototypes that obtain the best results although there are other prototypes with similar values. In general, prototypes computed using the *Graduated Assignment* obtains better results than the *Hierarchical* and *Genetic synthesis*. Considering values on Table 6.2 and Table 6.4 (KNN), we can conclude that the probabilistic prototypes are slower but obtain greater *precision*. And considering values on tables Table 6.3 and Table 6.5

(*similarity*), we realise that the fact that some queries only explore the root node penalises the obtained *precision*.

Precision (k=3)	**Synthesis**	**COIL**	**LETTER-LOW**	**LETTER-HIGH**	**GREC**
Set Median	-	0.46	0.76	0.42	0.48
Generalise Median	Hierarchical	0.37	0.62	0.34	0.22
Generalise Median	Genetic	0.11	0.16	0.22	0.34
Generalise Median	Graduated Assignment	0.32	0.94	0.46	0.32
Closure Graph		0.62	**0.98**	0.38	**0.59**
FORG		0.58	0.86	0.42	0.57
SORG		**0.65**	0.81	**0.50**	0.37
FDG		0.48	0.84	0.44	0.53

Table 6.4. Precision on nearest neighbour queries.

Precision ($d_{max} = 0.6 \cdot D_{max}$)	**Synthesis**	**COIL**	**LETTER-LOW**	**LETTER-HIGH**	**GREC**
Set Median	-	0.75	0.99	0.99	0.78
Generalise Median	Hierarchical	0.81	0.99	0.99	**0.98**
Generalise Median	Genetic	0.89	**1**	0.99	0.92
Generalise Median	Graduated Assignment	**0.99**	0.99	0.99	0.97
Closure Graph		0.75	0.62	0.74	0.77
FORG		0.75	0.62	0.74	0.77
SORG		0.75	**1**	**1**	0.77
FDG		0.75	0.82	0.78	0.77

Table 6.5. Precision on similarity queries.

Recall (k=3)	**Synthesis**	**COIL**	**LETTER-LOW**	**LETTER-HIGH**	**GREC**
Set Median	-	0.26	0.58	0.32	0.50
Generalise Median	Hierarchical	0.24	0.48	0.26	0.22
Generalise Median	Genetic	0.06	0.12	0.18	0.34
Generalise Median	Graduated Assignment	0.18	0.72	0.36	0.34
Closure Graph		0.38	**0.76**	0.30	**0.60**
FORG		0.38	0.66	0.32	0.58
SORG		**0.40**	0.62	**0.38**	0.38
FDG		0.30	0.64	0.32	0.54

Table 6.6. Recall on nearest neighbour queries.

Table 6.6 and Table 6.7 show the *recall* results. In general, probabilistic prototypes obtain greater *recall* than non-probabilistic ones, except in some cases. Note that in cases that the *access ratio* is 0.02, the *recall* is always 1.

This is because, if all graphs of the database are accepted, then the *recall* has to be 1 by definition.

Recall ($d_{max} = 0.6 \cdot D_{max}$)	**Synthesis**	**COIL**	**LETTER-LOW**	**LETTER-HIGH**	**GREC**
Set Median	-	**1**	0.99	**1**	0.99
Generalise Median	Hierarchical	**1**	**1**	0.98	0.90
Generalise Median	Genetic	0.92	**1**	**1**	0.96
Generalise Median	Graduated Assignment	**1**	0.98	0.98	0.90
Closure Graph		**1**	**1**	**1**	**1**
FORG		**1**	**1**	**1**	**1**
SORG		**1**	**1**	**1**	**1**
FDG		**1**	0.60	0.90	**1**

Table 6.7. Recall on similarity queries.

Finally, Table 6.8 summarises the results presented in the last 6 tables. Each value is the average of the eight corresponding values. Statistically best values are bolded. FORGs obtain the fastest queries (lower *access ratio*).Generalise Median (with *Graduated Assignment*) and SORGs obtain the greatest *precision*. *Set Median*, *Closure Graphs*, FORGs and SORGs obtain the greatest *recall*. Finally, *Generalise Median* (with *Graduated Assignment*) and SORGs obtain the best *F-measure*.

Prototype	**Synthesis**	**Access Ratio**	**Precision**	**Recall**	**F-measure**
Set Median	-	0.53	0.70	0.70	0.70
Generalise Median	Hierarchical	0.65	0.66	0.63	0.64
Generalise Median	Genetic	0.84	0.57	0.57	0.57
Generalise Median	Graduated Assignment	0.70	**0.74**	0.68	0.71
Closure Graph		0.30	0.68	**0.75**	0.71
FORG		**0.21**	0.66	0.74	0.70
SORG		0.81	**0.73**	0.72	**0.72**
FDG		0.29	0.64	0.32	0.67

Table 6.8. Average results of access ratio, precision, recall and F-measure.

Chapter 7 Cooperative Pose Estimation Based on Interactive Points Alignment

Throughout this thesis we have explored the way to improve the *Error-Tolerant Graph Matching* performance taking advantage of the feedback provided by an oracle. On the one hand, it has been explored different active query strategies to obtain the perfect correspondence between two graphs after few interactions (Chapter 3) and on the other hand, it has been proposed a loss function to find the parameters such that the *hamming distance* between the automatic/suboptimal and the ground-truth correspondence is minimized (Chapter 4 and Chapter 5).

The present chapter delves into the idea of using an interactive feedback to improve automatic results, bringing it to a different domain. In this chapter we focus on the implementation of interactivity in a framework of cooperative robotics.

Given a fleet of autonomous robots performing a cooperative task, it is crucial for the robots to share their relative position. If the site has not been explored before, auto localise each robot through landmarks is not possible. Moreover, not always the GPS information is available or it has the desirable accuracy. Our framework is composed of a fleet of robots that have 2D and 3D cameras, a human coordinator and a human-machine interface. 3D-images are used to automatically align them and deduce the relative position between robots. 2D-images are used to reduce the alignment error in an interactive manner. A human visualises both 2D-images and the current automatic alignment and imposes a new alignment through the human-machine interface. Since the information is shared through the whole fleet, robots can deduce the position of other ones that do not visualise the same scene. Practical evaluation shows that in situations that there is a large difference between images, the cooperative and interactive processes are crucial to achieve an acceptable result.

7.1 Introduction

In recent years, interaction between robots and humans and also cooperation between robots has increased rapidly. Applications of this field are very diverse, ranging from developing automatic exploration sites [160] to using robot formations to transport and evacuate people in emergency situations [161]. Within the area of social and cooperative robots [162], interactions between a group of people and a set of accompanying robots have become a primary point of interest [163].

One of the low level tasks that these systems have to face is the automatic pose estimation. If the information of GPS is not available or its accuracy is not enough, one of the usual methods is to localise the robots through detecting landmarks or identifying scenes previously classified. The problem of comparing or aligning two images is usually called *image registration* in the computer vision research field. Image registration tries to determine which parts of one image correspond to which parts of another image. This problem often arises at the early stages of many computer vision applications such as scene reconstruction, object recognition and tracking, pose recovery and image retrieval. Therefore, it has been of basic importance to develop effective methods that are both robust in the sense of being able to deal with noisy measurements and in the sense of having a wide field of application.

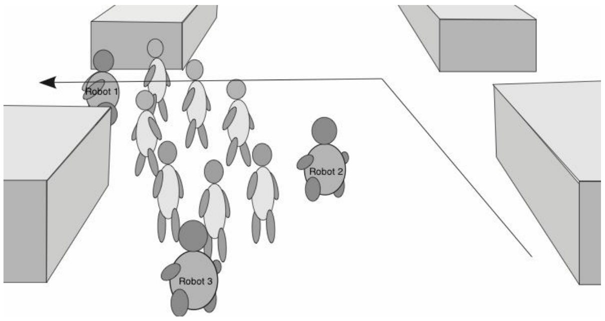


Figure 7.1. Three robots performing guiding tasks. Robots are located to fence the group.

We present an interactive and cooperative method to deduce the relative pose of each robot with respect to the rest of the fleet. The proposed model is part of a larger project in which social robots guide people through urban areas [163] and they have tracking abilities [166]. Figure 7.1 represents three robots performing guiding tasks in an indoor environment. Robots fence the visitor group to force them to follow a specific tour. Robots need to work in a cooperative manner to keep a triangular shape in which people have to be inside. In these cooperative tasks, it is crucial to have a low-level computer vision task such that images extracted from the three robots are properly aligned to correctly deduce their relative pose. In the proposed environment, there is a human that, through our human-machine interface, gives orders to the robots and controls their tasks. Robots have embedded 2D and 3D cameras [167] and the human can visualise the 2D cameras.

7.2 Image Registration

The three typical steps involved in the solution of the image registration problem are the following [175]. First, some salient points are selected from both images. Second, a set of tentative matches between these sets of points is computed together with the image alignment. And third, a process of outlier rejection that eliminates the spurious correspondences can further refine these tentative matches and the initial alignment.

Salient points, which play the role of parts of the image to be matched, are image locations that can be robustly detected among different instances of the same scene with varying imaging conditions. These points can be corners (intersection of two edges) [176], maximum curvature points [177] or isolated points of maximum or minimum local intensity [178]. There is an evaluation of the most competent approaches in [83]. When salient points have been detected, several correspondence methods can be applied that obtain the alignment (or homography) that maps one image into the other [179], discards outlier points [180] or characterises the image into an *attributed graph* [79, 80]. Typically, these methods have been applied on 2D images but recently, 3D shape retrieval methods have appeared [181].

The main drawback of image registration methods is that their ability to obtain the correspondence parameters strongly depends on the reliability of the initial tentative correspondences. Moreover, it is needed to jointly estimate the image alignment parameters and correspondence parameters.

Considering the alignment parameters, there are two basic strategies. The first one is to consider a rigid deformation and the second one is to consider non-rigid deformation. In the first case, it is assumed the whole image (and so, the extracted salient points) suffers from the same deformation and so the image alignment parameters are applied equally to the whole salient points or image pixels. Some examples are [79, 182, 183, 32]. In the second case, each salient points suffers a different projection and there are different alignment parameters applied to each salient point or image region. Some examples are [184, 185]. Usually, the rigid strategy is applied to detect objects on outdoor images in which the deformation is mostly due to the change of the point of view. The non-rigid strategy is mostly applied to object detection or matching in medical or industrial images due to it is assumed objects suffer from deformations although the point of view is the same.

7.3 Human Interactivity

Humans are very good at finding the correspondences between local parts of an image regardless of the intrinsic or extrinsic characteristics of the point of view. Human interactivity on image registration has been applied on medical images [186, 187, 188] and two systems have been patented [189, 190]. These papers and patents are really specific on some medical environments and for this reason cannot be applied on our problem. In [186], they show a comparison of 3-D images on MRI-SPECT format and they concretise on images from the brain. In [187], authors present a method to validate the 3D position given 3D medical images. Finally, in [188], the aim is to solve the registration problem given similar medical images extracted from different sensors or technologies. Patent [189] defines a system for registration thorax X-Ray images such that it does not depend on bony structures. Finally, patent [190] defines a multi-scale registration for medical images where images are first aligned at a course resolution, and subsequently at progressively finer resolutions; user input is applied at the current scale. Another usual application of human interaction is semi-automatic video annotation [191].

Current automatic methods to extract parts of images and their correspondences in non-controlled environments are far away of having the performance of a human. Figure 7.2 shows two images. In each image 50 salient points have been extracted by method [77]. Outlier detector [180] has considered 43 salient points where outliers and only 7 where inliers. The correspondence detector has missed 6 of the 7 points (red lines) and only has hit 1 point (green line). This is because of the large differences between both images and more precisely, due to the lack of ability of the initial correspondence detector to find a good initial correspondence.

For this reason, in this chapter, we propose a semi-automatic method in which humans can interact into the system when it is considered the quality of the automatically found correspondences is not good enough and then they impose a partial and initial correspondence between some local parts of two scenes.

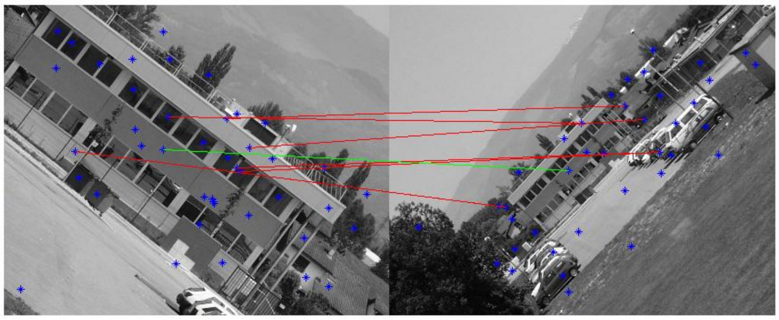


Figure 7.2. An example of automatic registration image where only one point has been properly matched (green line) and 6 points have been improperly matched (red lines).

A human-machine interface (HMI), also named human-computer interface, provides more natural, powerful, and compelling interactive experiences. For decades, HMI has been an active research field closely related to new technology advances. Paper published in [194] provides an interesting review of the current trends and key aspects of HMI, for instance in non-desktop computing. Some of the latest applications have been natural language understanding [195], semi-supervised database indexing [196], image segmentation [197, 198] or handwritten text transcription [94] between others. Finally, human interactivity specifically applied to pattern recognition has been considered in [75].

Considering the visual presentation of the HMI, three main approaches have been proposed in the literature: (a) side-by-side views [199, 200], (b) superimposed or merged views [201, 202] and (c) animations [203, 204]. These approaches are often complemented with techniques for highlighting the correspondences between points such as lines that go from one set of points to the other [199, 205] or elements colour coding [206, 207]. We have used the side-by-side view (Figure 7.2).

7.4 Overview of the proposed model for pose estimation

In this chapter, we concretise on how to deduce the relative pose of our robots in an interactive and cooperative manner and the experimental section only tests this aspect. The other technical and theoretical and technical aspects are not intended to be discussed in this thesis.

We say it is an interactive method because a human helps the image processing modules incorporated on the robots to automatically solve the 3D registration problem when it is necessary. Figure 7.3 shows part of our human-machine interface. We can visualize the 2D cameras of robots 1 and 2 and the correspondences imposed by the user. Both robots go on the pavement and Robot 1 follows Robot 2. Since the 2D and 3D cameras in each robot are calibrated, the imposed correspondence in the 2D domain is translated to a correspondence in the 3D domain and so this interaction influences over the obtained 3D alignment and the relative pose is recomputed.

Thanks to this interaction, the accuracy of the pose estimation of the whole feet of robots increases. This type of interaction is completely different from the ones presented in [160, 161, 163] since in those cases, the human interaction is performed in a higher level. For instance, in [160, 161], the interaction is based on imposing orders such us “move straight ahead” or “go up stairs”. In [163], the orders are “follow this person” or “bring me to the exit”. In [171], the interaction is used to learn the matching process. Nevertheless, our experience has shown us that in most of the cases, robots cannot perform these orders due to they cannot solve the low-level registration problem. Therefore, human interaction in this low-level task, that it is really easy for humans, frequently makes not necessary to interact on higher level tasks in which the interaction is more complicated do to the need of having more knowledge of the current situation such as position of other robots, automatically built map of the environment or current position of other objects.

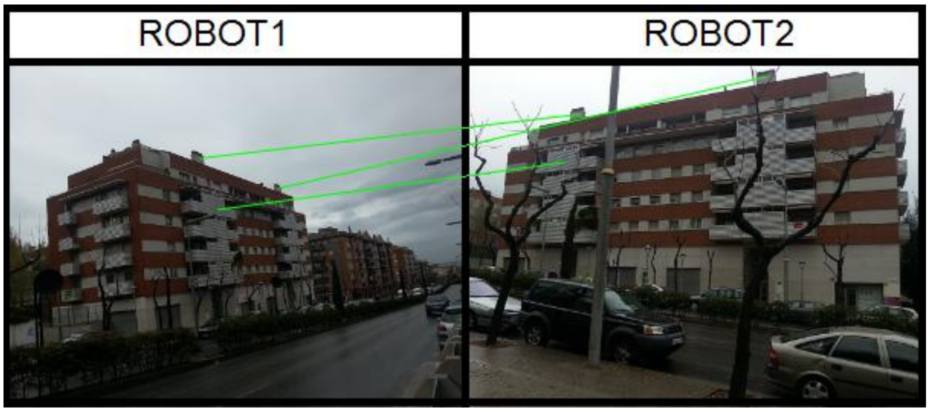


Figure 7.3. Screen shot of our human-machine interface with 2D images of Robots 1 and 2.

Besides, it is a cooperative model since the relative pose of two robots is deduced through the other robots when these robots do not share any part of the scene they visualise. Figure 7.4 shows the position of three robots in the 2D plane. Robot 1 and 2 can theoretically deduce their relative position through 3D image registration [172] but this is not possible between Robot 1 and 3 since they do not share any part of the 3D image. This problem is solved through the cooperation of the robots. Robot 2 deduces its relative position with respect to Robot 3 and shares this information with Robot 1.

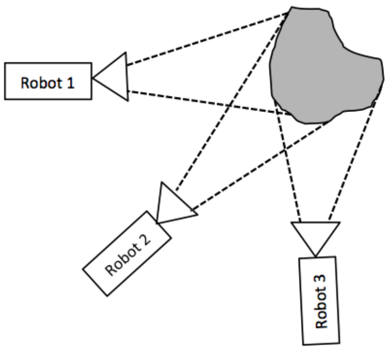


Figure 7.4. Three robots visualising the same scene.

Figure 7.5 shows a schematic view of our method based on an *Interactive Pose Estimation* module and a human-machine interface. The input of the general system is the 2D and 3D images of all robots and the output is their relative pose and the regression error (as it does the GPS). We are interested on minimising the pose error but in a real application, we cannot deduce this error since we do not have the *ground-truth pose*, but we can obtain the regression error between aligned images. We know the lowest is the regression error, the best is the pose estimation. Then, we assume there is a direct dependency between the regression error and the pose error. Thus, reducing the regression error, the system tends to reduce the pose error.

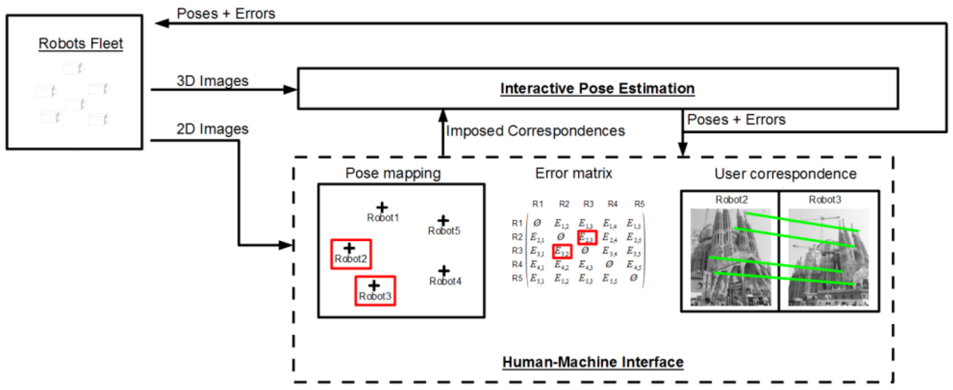


Figure 7.5. Basic scheme of our method composed of an Interactive Pose Estimation module and the human-machine interface.

On the one hand, the human-machine interface receives from the fleet of robots the 2D-images and from the *Interactive Pose Estimation* module the current poses of the robots in the feet and the deduced error of these poses. The human-machine interface outputs the user impositions to the *Interactive Pose Estimation* module and it does not output any information to the fleet of robots. On the other hand, the *Interactive Pose Estimation* receives from the fleet of robots the 3D images and returns to it the poses estimation and the regression errors.

The proposed human-machine interface is as follows. On the left side of it, the user visualises the deduced current pose of the fleet. On the middle of the interface, it is shown the regression error between any combinations of robots' images. We make this matrix to be symmetric since we introduce in cell $[i,j]$ and in $[j,i]$the same values, which are the last regression error while deducing the position of the i^{th} robot with respect to the j^{th} robot or vice versa. The maximum regression errors are highlighted on bold to attract the attention to the user. On the right side of it, the user visualises the 2D images of two manually selected robots together with the imposed correspondences. The user can visualise any of the combinations of 2D images by selecting one of the cells in the middle of the human-machine interface. Then he can update the imposed correspondence by erasing or creating mappings between points. The two robots in the left panel and the regression errors in the central panel that correspond to the current images in the right panel are highlighted in red. If the regression error between two robots is higher than a threshold, we assume that the corresponding images do not share any salient points and then the regression error is automatically hided (marked with a hyphen). Nevertheless, if the user considers that these robots really visualise the same part scene, he can select those robots. Note the 2D images are not an input of the *Interactive Pose Estimation* module but they are used to be visualised by the user and impose the points correspondences.

A preliminary version of this method was presented in [170, 173], in which, was presented a simple interactive method to estimate the homography between two 2D images. There were no 3D images and so the poses of robots were not deduced. When we put into practice our system, we realised most of the cases that robots did not correctly react to the humans orders where due to they did not solve appropriately the low-level image registration problem. Therefore, putting the human interaction into the image alignment problem, most of these non-correct robot reactions are solved. Nevertheless, technology tends to make systems to run as much autonomous as possible. For this reason, the main weakness of our method is that robots are less autonomous. Better registration methods are going to be discovered, and then, less need of human interaction is going to be needed. A similar interaction method was presented in [174]. In that case, there is only one robot and the human decides if a selected part of the image is a human's face and in the case that it is, imposes the name of the person.

Finally, Figure 7.6 shows the general scheme of one relative pose estimation. First, a pair of robots is selected according to the scheduler strategy. Then, the *Robot Interactive Homography Estimation* module (section 7.5) estimates the homography between the selected robots according to the 3D images of the robots cameras and the user imposed correspondences. Then, the *Cooperative Pose Estimation* module (section 7.6) updates the relative poses according to the previously estimated homography and the current relative poses. This process is recursively executed by the task scheduler.

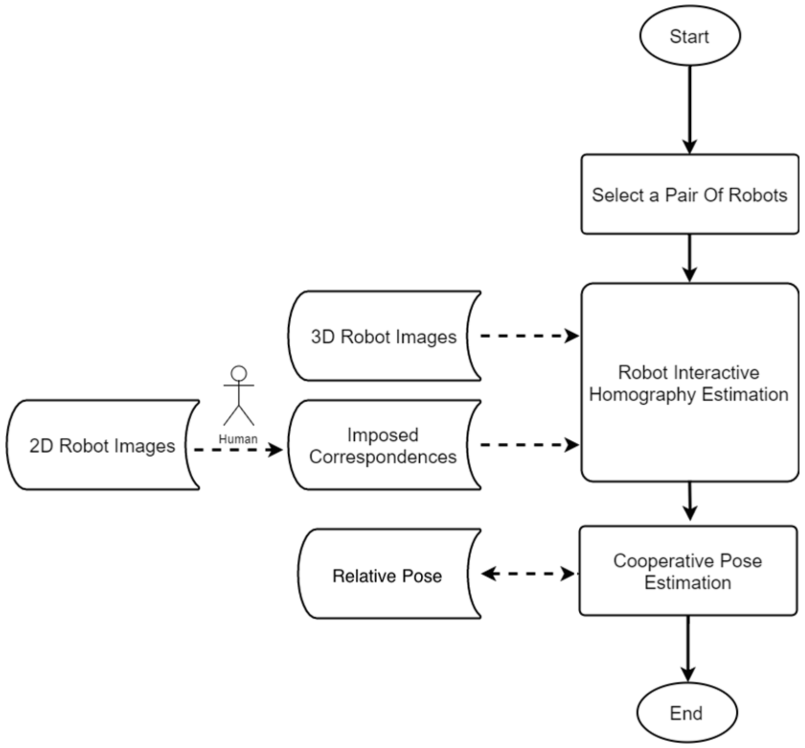


Figure 7.6. General scheme of one relative pose estimation.

7.5 Interactive Pose Estimation module

Figure 7.7 shows our *Interactive Pose Estimation* general scheme, which is the upper module in Figure 7.5. The aim of the *Scheduler* is to keep updated the consistency of the relative position information between the robots. To achieve this, it is responsible for selecting the pairs of robots to deduce their relative pose using an implementation of the weighted round robin algorithm [208]. This selection depends on the time elapsed since the last pose update, the known current pose, the regression errors and also the correspondences imposed by the user. The scheduler also provides to the *Robot Interactive Homography Estimation* module the homography $H^*_{i,j}$, deduced from the user imposed correspondences. Then, the *Robot Interactive Homography Estimation* module (Figure 7.8) deduces the relative homography $H_{i,j}$ of the i^{th} robot with respect to the j^{th} one. It takes into consideration the imposed correspondences between these robots if they are any, and also the 3D-image alignment $H_{i,j}^{*}$. Moreover, the module returns a regression error of the obtained projection $Error_{i,j}$. As we have seen in Figure 7.5, this regression error is visualised at the human-machine interface to help the user to decide which pairs of robots need some point mappings to be manually imposed. The *Cooperative Pose Estimation* module deduces the pose of the whole fleet taking into consideration these homographies and the regression errors. To do so, it keeps a graph in which nodes represent robots and the weight of edges are the regression errors. Thus, to deduce the relative pose between two robots, the classical *Dijkstra* algorithm is performed (section 7.6).

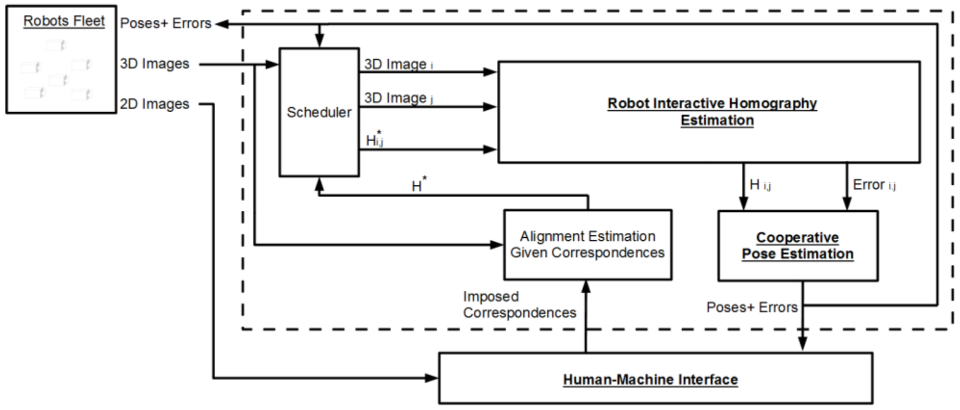


Figure 7.7. Interactive Pose Estimation module bounded by the dashed rectangle.

The *Alignment Estimation Given a Correspondence* module deduces a homography $H_{i,j}^{*}$ between these images taking into consideration the human's correspondence proposal. It is based on the *Direct Linear Transformation* algorithm [209, 210] that solves a set of variables from a set of similarity relations. It obtains a matrix homography (or linear transformation) $H_{i,j}^{*}$ which contains the unknown parameters to be solved. In [170], authors used this algorithm to obtain a homography between two sets of 2D points. In this scheme, the same method is used but with 3D points.

We assume the transformation between two 3D images (and so, their 3D points salient points) can be modelled as an affine transformation in the 3D space. Then homography $H_{i,j}^{*}$ is computed as follows,

$$H_{i,j}^{*} = \begin{bmatrix} & 0 & 0 & x \\ 0 & b & 0 & y \\ 0 & 0 & c & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $a = S \cdot cos(\beta) \cdot cos(\gamma)$, $b = S \cdot cos(\alpha) \cdot cos(\gamma)$ and $c = S \cdot cos(\alpha) \cdot cos(\beta)$. Parameter S is the scale and α, β and γ are the three orientation angles of one robot with respect to the other. Besides x, y and z is the translation of one robot with respect to the other. The n correspondences imposed by the user between the two 3D points are represented by $(x_i^k, y_i^k, z_i^k) \to (x_j^k, y_j^k, z_j^k)$, being $1 \leq k \leq n$. In this case, the error function is usually estimated as follows:

$$E(a, b, c, x, y, z) =$$

$$\sum_{k=1}^{n} \left[\left[a \cdot x_i^k + x - x_j^k\right]^2 + \ [b \cdot y_i^k + y - y_j^k]^2 + \ [c \cdot z_i^k + z - z_j^k]^2 \right]$$

Then, values (a, b, c, x, y, z) are obtained in a minimisation process through deriving the function; $\frac{\delta E}{\delta a} = 0$, $\frac{\delta E}{\delta b} = 0$, $\frac{\delta E}{\delta c} = 0$, $\frac{\delta E}{\delta x} = 0$, $\frac{\delta E}{\delta y} = 0$ and $\frac{\delta E}{\delta z} = 0$. Note that to obtain an exact value of the angles and translations, a minimum of $n = 6$ is needed. Nevertheless, thanks to the *Cooperative Pose Estimation* module and the fact that the 3D points are shared in several images, these values can be estimated in fewer points per image.

If the user does not impose correspondences between two specific images, the module returns the identity matrix for this pair of images.

Figure 7.8 shows the *Robot Interactive Homography Estimation* module. The *Salient Point Extractor* module obtains a pair of sets of 3D-points [172] given a pair of 3D-images. The set of points $3Dpoints_i$ are projected to $H_{i,j}^{*}$ to obtain the same set of points but referenced in the same coordinate system than the set $3Dpoints_j$ and considering the human's correspondence proposal. We call this set of points $3Dpoints_i^*$. This process provides a better initial alignment to the *Iterative Closest Point* algorithm (ICP) [179] to find a final alignment of both sets of points through homography $H'_{i,j}$. The regression error $Error_{i,j}$ is defined as the sum of square distance between points in $3Dpoints_i^*$ and the projected points $3Dpoints_j \cdot H_{i,j}^*$. The module returns homography $H_{i,j} = H_{i,j}^* \cdot H'_{i,j}$.0

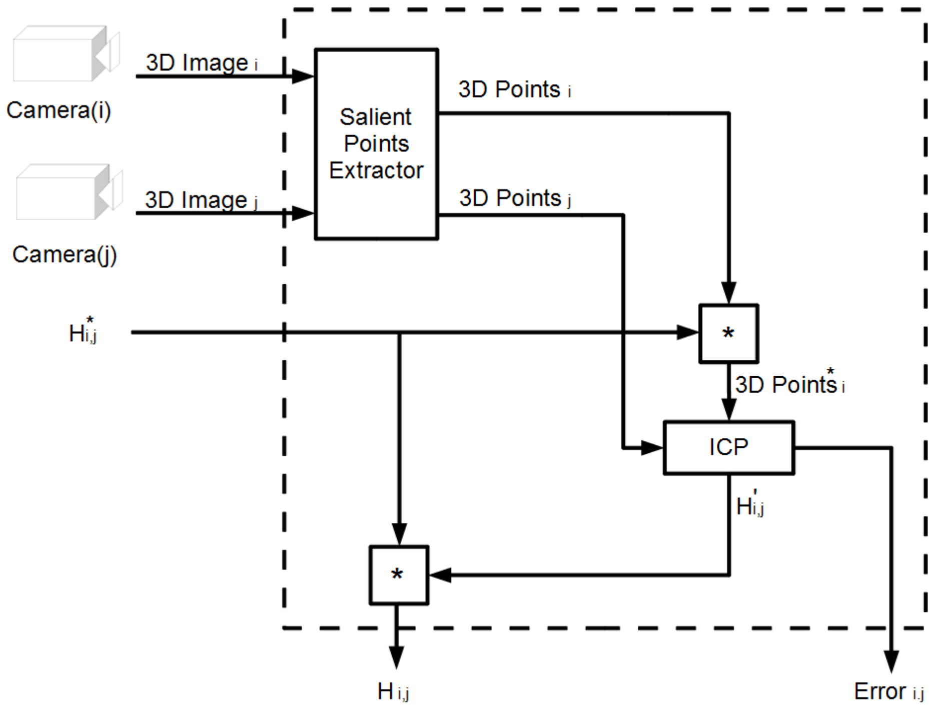
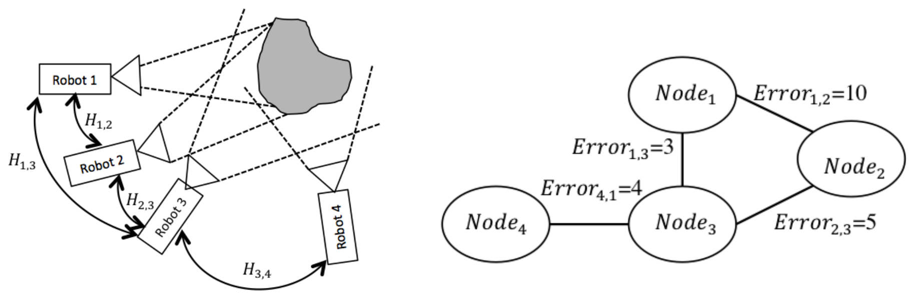


Figure 7.8. Robot Interactive Homography Estimation module.

7.6 Cooperative Pose Estimation

Figure 7.9.a shows a scene where four robots visualise an object and there is a high overlapping between images of the first three robots but Robot 4 only shares part of the image with Robot 3. Initially, they do not know the relative position of the other robots (remember, robots do not have any system to locate themselves in the scene) and therefore they do not know which robot is close to another. Robot 1 can deduce the relative position of Robot 2 directly (through homography $H_{1,2}$) or indirectly through Robot 3 (applying the product of homographies $H_{1,3} \cdot H_{3,2}$). How do we know which is the best way to deduce the relative position? The method we propose obtains the final homography between Robot 1 and Robot 2 that achieves the lower regression error. In this example, $Error_{1,2} = 10$, $Error_{1,3} = 3$ and $Error_{3,2} = 5$, then $Error_{1,2} \geq Error_{1,3} + Error_{3,2}$ and for this reason, we decide the homography between Robot 1 and Robot 2 is computed as $H_{1,3} \cdot H_{3,2}$ instead of $H_{1,2}$ although we see in the scene that Robot 1 is closer to Robot 2 than Robot 3. Since each homography can be achieved by several combinations, we select the combination that minimises the sum of regression errors. Moreover, the Robot 1 and 2 can only deduce the position of Robot 4 through Robot 3. Thus, the homography between Robot 1 and Robot 4 is computed as $H_{1,3} \cdot H_{3,4}$ and the homography between Robot 2 and Robot 4 is computed as $H_{2,3} \cdot H_{3,4}$.



a. b.

Figure 7.9. a. Four robots visualising an object and the homographies that transform their coordinate systems into the coordinate systems of the other robots. b. Graph of the current regression errors estimated by the Iterative Closest Point algorithm.

To do so, we use a graph that it is dynamically updated when new regression errors are obtained from the Iterative Closest Point algorithm (Figure 7.9.b). Nodes represent robots and the attributes on the edges are the last obtained regression error. Thus, a combination such as $H_{1,3} \cdot H_{3,2}$ is defined as a path from Robot 1 to Robot 2 composed of two steps $Node_1 \rightarrow Node_3 \rightarrow Node_2$. In general, the homography between Robot i and Robot j is decided through the less costly path in the graph from the first robot to the second one. That is, $H_{i,j} = H_{i,a} \cdot H_{a,b} \cdot H_{b,c} \cdots H_{k,j}$ if $Node_i \rightarrow Node_a \rightarrow Node_b \rightarrow Node_c \rightarrow \cdots \rightarrow Node_k \rightarrow Node_j$ is the path from $Node_i$ to $Node_j$ such that

$Error_{i,a} + Error_{a,b} + Error_{b,c} + \cdots + Error_{k,j}$ obtains the minimum value. We use the well-known *Dijkstra* algorithm [211] to obtain this minimum path between two nodes of the graph.

7.7 Experimental Evaluation

The practical evaluation is explained in the following subsections. In “Sagrada Família” database section of the Databases annex, we describe the experimental setup of this chapter. In the following two sections of this chapter, we analyse the robots' position error and the runtime. Due to the nature of our database we know the real position of the robots. Thus, the position error is computed as the Euclidean distance between the correct position and the deduced one. In 7.7.1, we analyse the influence of the human interaction and we do not have used the *Cooperative Pose Estimation* module (all used paired images have some 3D-points in common). In 7.7.2, we have considered the whole system and dataset.

7.7.1 Interactive Pose Estimation

Figure 7.10 shows the robot position error in meters of the system with respect to the number of interactions and the level of separation between images (from 5 to 45). In these first experiments, the cooperative module has not been used. As it is supposed to be, far away are the images, larger is the error since less 3D-points are shared (see “Sagrada Família” database section of the Databases annex) and also larger is the distortion between images. Moreover, when only one or two interactions are imposed, only translations can be deduced in the alignments, for this reason, the robot position error reduction is not so important. It is clear that with three interactions, the error is drastically reduced independently of the level of separation between paired images. This is because an affine homography can be deduced. Finally, when more than three interactions are imposed, the error is only slightly reduced.

Given the results of Figure 7.10, we could recommend to the user to interact a maximum of three times per pair of robots through all pairs of robots instead of interacting more than three times in some few pairs of robots and keeping some pairs of robots without interaction. Nevertheless, since we assume robots are moving and therefore images are constantly updated, we only have the reliable information of the current regression error. From Figure 7.10, we see it is better to interact on the pairs of robots that have the largest regression errors because in these cases the human interaction accentuates the regression error's decrease. We highlight the need of this human's interaction through painting in bold the pairs of robots with the largest regression errors in the human-machine interface (Figure 7.5).

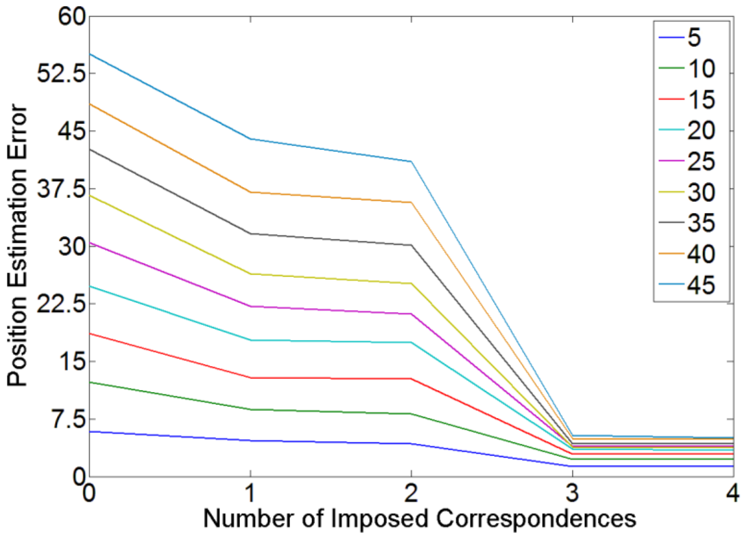


Figure 7.10. Mean robot position error in meters with respect to the number of interactions and the level of separation.

A crucial aspect of the cooperative robotic systems is the ability to keep the information updated in time. In this case, we need to know the pose of the fleet of robots in real time. The most costly process is the ICP algorithm (Figure 7.8) that it is performed each time a new image arrives, independently if there is human interaction or not. It has a quadratic worst cost with respect to the number of elements in the images (note the huge number of points shown in the Figure A.9 of the Databases annex). Another costly process is the *Alignment Estimation Given the Correspondences* (Figure 7.7), which it is performed through the Direct Linear Transform algorithm. Nevertheless, it is only performed with the points that the human has interacted and, as explained before, we recommend a maximum of three interactions. For this reason, the runtime of this algorithm is almost negligible. Another process to be considered is the *Cooperative Pose Estimation* based on the *Dijkstra* algorithm (Figure 7.7). This algorithm has a quadratic cost with respect to the number of nodes in the graph [211]. Since in our framework the number of robots (which it is the number of nodes in the graph) is lower than ten, again, the runtime does not affect so much at the general runtime since it has the same complexity than the ICP but with really less points. For this reason, we have only analysed the runtime of computing the ICP.

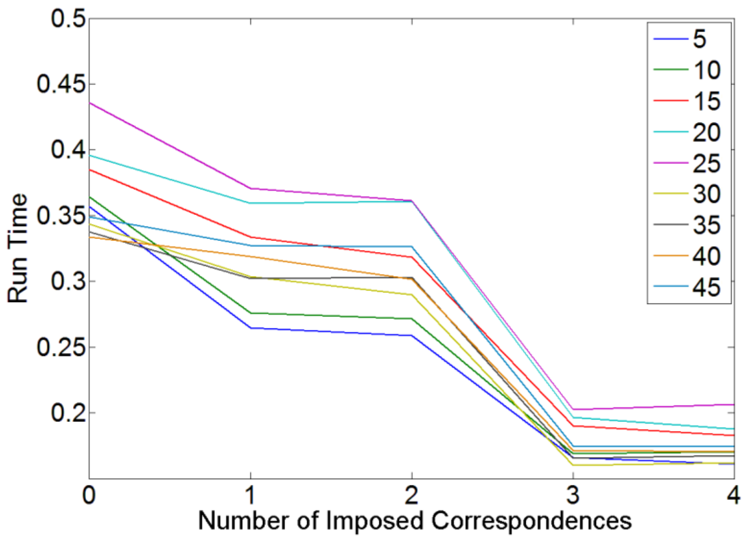


Figure 7.11. Runtime of the ICP module in seconds with respect to the number of interactions and the level of separation (Matlab, i7 950, 3.07 GHz, 6 GB RAM, Windows 7).

Figure 7.11 shows the runtime in seconds of the ICP with respect to the number of interactions and the level of separation between images (from 5 to 45). ICP is a regression method that finds the solution accurately and faster when the initial alignment is more congruent to the affine matrix. Considering the human's interactions, more interactivity, more accurate is the homography matrix $H^*_{i,j}$ deduced in the *Alignment Estimation Given a Correspondence* module and so more easy is the task of the ICP to find the homography $H'_{i,j}$.

Finally, it is important to consider that previous human interactions positively influence on the pose accuracy and runtime of the following non-interactive estimations. That is, the following estimations tend to be more accurate and faster deduced.

7.7.2 Cooperative pose estimation

Table 7.1 shows the robot position error of the whole system given the separation between robots of 50, 100, 150 and 200 meters and considering that the human has not interacted, interacted once or three times in all images pairs that have three or more common points. When the separation between robots increases, so it does the error. The error is really large in the cases that there is no interaction. This column is useful to show the level of difficulty of the *SAGRADA_FAMILIA* database. We realise the automatic method fails on deducing the homography between images and so the position of the robots. We believe this is because the images are so distant that they share really few points and the point of view of these points are really distinct (Figure A.9 of the Databases annex).

Separation between robots (meters)	Number of imposed correspondences		
	0	1	2
50	38.33	28.13	7.09
100	83.36	63.91	14.65
150	144.28	131.60	22.41
200	168.15	158.07	24.79

Table 7.1. Robot position error in meters obtained using the whole system.

Separation between robots (meters)	Number of imposed correspondences		
	0	1	2
30	1.25	1.53	1.75
60	1.03	1.97	2.47
120	1	2.19	3.61
180	1.19	2.02	4.47

Table 7.2. Average number of required steps obtained using the whole system.

Moreover, we can see that with one interaction, the system achieves better results but the reduction of the error is not so significant. With three interactions, the system obtains an important reduction with respect to non-human interaction case. Similarly to the errors shown in Figure 7.10, there is not an important reduction of the error with more than 3 interactions. We conclude the human interaction is crucial in this database to reduce the position error. Moreover, more than 3 interactions are not needed.

Finally, Table 7.2 shows the average number of required steps to deduce the cooperative pose estimation module and computed by the *Dijkstra* algorithm. As it is expected, the more separated are the images, the more steps are needed. In general, the number of steps tends to increase when the number of interactions increases. This is because the regression errors computed by the ICP tend to decrease when interactions increase (section 5.2) and then the *Dijkstra* algorithm finds larger paths but with lower costs.

Chapter 8
Conclusions & Future Work

8.1 Contributions of this thesis

The breadth of the topics covered in this thesis make it necessary to summarize the discussion about the experimental results separately. The following subsections correspond to conclusions drawn from experimental results. Chapter 4 and Chapter 5 are discussed together (in section 8.1.3).

8.1.1 On the relevance of structural dissimilarity metrics on Graph Edit Distance

In Chapter 2 four specific definitions for node local structures have been presented, namely *Node*, *Degree*, *Star* and *Mesh*. For the computation of both *Node* and *Degree* no additional algorithm is required. Yet, for *Star* and *Mesh* structures four and two particular algorithms are needed respectively. In total, there are eight different cost models, viz. *Nodes*, *Degree*, *Hungarian*, *Planar*, *Hausdorff*, *Greedy*, *Eigenvector* and *Pagerank* and it has been shown the relevance of the each one to estimate the structural dissimilarities on *Graph Edit Distance* for graph classification while using the SFBP and the recognition ratio and runtime in different databases with different configurations of edit costs. We conclude that when the structural information is not so relevant, *Node* cost model achieves similar accuracies than the other ones but it is really much faster. Besides, when the structural information and nodes' attributes are equally important, the *Degree* cost model can be a good option because it presents a good balance between accuracy and runtime. In the cases that the attributes of the nodes are not important or graphs are sparse, the cost models that really takes into consideration the *Mesh* of the graph, such as *Eigenvector* and *Pagerank*, are the ones that achieve the best accuracies. *Hungarian* and *Planar* solvers for the *Star* local structure are good options when the local structural information is useful and the main goal is the accuracy in detriment of the runtime. Finally, we could say that *Hausdorff* and *Greedy* solvers for the *Star* local structure are faster than *Hungarian* and *Planar* and more accurate than *Degree* and *Node*. Nevertheless, in a real application, we need to study in depth the type of data we have or test the behaviour of these centralities in a validation process to select the best cost model in order to maximise the specific goals (accuracy or runtime).

8.1.2 Using active queries for interactive Error-Tolerant Graph Matching

In Chapter 3 it has been presented an active and interactive algorithm for graph matching. In relation to the active part, there are 6 different strategies. These strategies are based on classical active machine learning but they are applied to the case of searching for the best mapping between nodes. Moreover they are based on the probability matrix between nodes that some sub-optimal algorithms iteratively compute to find the best correspondence. In relation to the interactive part, it has been presented a method to modify the edit costs so that the original values are updated to force the correspondence to be similar to the one proposed by the oracle. Given that our new algorithm only updates the costs between nodes and edges and reads the probability matrix, it is not necessary to modify the code of the well-known graph matching algorithms.

The method is proposed to be used in applications where obtaining the perfect correspondence is crucial. For instance, medical applications in which the doctor wishes to compare two images and find the mappings between their parts. In applications where there is a need for comparing the unclassified graph with a large number of graphs in a database (for instance, Automatic Fingerprint Identification System), the oracle's feedback could penalise the run-time of the system. Therefore, there is a compromise between run time and performance. Finally, there are applications with huge graphs (for instance, social networks or ontologies). If the oracle selects some of the nodes to be mapped, then, the graph matching algorithm could be greedier and solve the problem faster.

Experimental evaluation shows that MGN is the best strategy. However, we have to ensure that the extracted features and the definition of the matching cost function is concordant with what the oracle sees and believes.

8.1.3 Learning edit cost for graph matching

In Chapter 4 it has been presented a model for automatically learn the substitution weights (importance of each attribute) on nodes and edges while solving the graph matching problem. The importance on the attributes is gauged by some weights on the graph distance function, which are usually manually validated. More concretely, the aim of Chapter 4 is to present an optimisation function to find the weights such that minimize the distance between the ground-truth correspondence and the automatically-found correspondence (*hamming distance*).

As we have shown, the *hamming distance* is a step function depending on the weights and classical optimisation techniques based on the gradient cannot be applied due the value of the *hamming distance* at each region is constant in its domain. Some literature exists that solve this problem using a different loss function based on external variables, that is, to validate the solution with features not used to compute the graph distance (for instance, the point position). Nevertheless, this approach requires the premise that the ground-truth correspondence must be consistent of these external variables (for instance, the oracle wants to minimise the point position error). If we do not want to be restricted to this premise, we need to define a loss function that only depends on the correspondence cost. In this chapter, we show that the costs distance between the ground-truth and the automatically obtained correspondence is a good loss function. That is, there is a relation between the costs distance and the *hamming distance* functions.

It is also shown that *Class of Costs* (regions in the weight space with the same optimal correspondence) are radial sectors that have their origin at point $w = \vec{0}$. Thus, the predictor function is a radial function. For this reason, it has been proposed a regularization term that assures any correspondence is restricted but the optimisation process does not tend to obtain the trivial solution $\vec{0}$.

Experimental results show that in some cases we obtain a higher improvement than the compared method while learning the weights.

On the other hand, in Chapter 5 it has been defined a method to automatically establish the *insertion* and *deletion* edit costs. The aim is that, when a new database is presented to do research on different topics related to machine learning, these costs are computed and put together with the set of *attributed graphs* and so, the whole experiments are performed with the same pair of costs that have been shown that minimise the *hamming distance* between the automatically obtained correspondence and a ground-truth correspondence.

There are some structural pattern recognition applications that a human can easily define a ground-truth correspondence, for instance, we are very good at finding the correspondences between local parts of two images or characters regardless of the intrinsic or extrinsic characteristics of the point of view. Nevertheless, in another ones (for instance, chemical structures) it can be very difficult to define it due to the complexity of the structure or the number of local parts. The method we present has sense in the cases were a reliable ground-truth can be defined.

8.1.4 Graph databases based on metric-trees of Graph-Class Prototypes

In Chapter 6 it has evaluated a graph indexing technique based on metric-trees and several *Graph-Class Prototypes*. Specifically, we have studied the behaviour of *Graph-Class Prototypes* as routing elements of metric-trees. Several papers have been published that compare the accuracy of the evaluated *Graph-Class Prototypes*.

The evaluation has been performed using four different datasets with different characteristics. We see, from the practical validation, that probabilistic prototypes seem to achieve better results on KNN queries. On the contrary, the *Generalise Median* together with the *Set Median* seems to give better results on *similarity* queries at the cost of giving a larger *access ratio*. Up to now, *Set Median Graphs* and *Closure Graphs* where the only prototypes used as routing elements of metric-trees. Therefore, a general conclusion of this work is that other existing *Graph-Class Prototypes*, such as the probabilistic ones (FDGs or SORGs), can also be successfully used as routing elements of metric-trees in graph databases. Nevertheless, there is not an important quality difference between probabilistic methods (expressed through the four commented measures).

Table 6.1 shows some *precision* results obtained from the same databases but any indexing structure was used. As a conclusion of these results, we may say the probabilistic methods (SORGs and FDGs) obtain the best average results. And also, *Set Medians* only have good *precision* when there is small variability between graphs of the same class (*LETTER-LOW*).

These results have to be seen as the ground-truth of our method. The aim of indexing the reference database with a metric tree is to speed-up the system. The reduction of the time needed to query each graph is represented by the access-ratio measure. If the *access ratio* is (for instance) 0.5 then the run time with the indexing structure is half of the original run time. Therefore, the *access ratio* have to be lower than 1. Nevertheless, when we introduce indexing structures (metric-trees) in which the distance between elements is computed by sub-optimal algorithms, the triangle inequality is not guarantee anymore and false rejections appear.

The last two columns of Table 6.8 summarise the quality of the database queries. A high reduction of the *access ratio* is not admissible if it is related to an important reduction of the *precision*. If we admit a maximum *precision* lost of 10%, then only the *Generalise Median* with *Graduated Assignment* can be used and the run time is 72% of the run time without indexing structure. In this case, the speed-up seems to be not really important. Nevertheless, if we increase the maximum *precision* lost of 20%, then FDGs may be a good option since the run time is approximately 1/3 of the original. Note that SORGs obtain *access ratio* greater than 1 and so, the indexing scheme is useless. The third first options seem to be not interesting since they have similar *access ratio* than *Generalise Median* with *Graduated Assignment* but with higher *precision* lost. Finally, we can conclude that the two admissible options are *Generalise Median* with *Graduated Assignment* and FDGs.

8.1.5 Cooperative pose estimation

When a fleet of robots has a task to perform in a collaborative way, one of the most important low-level tasks that it has to face is the pose estimation of all robots. Clearly, the imminent reaction of each robot directly depends on its pose and the relative pose of the other robots with respect to them. Besides, an inaccurate pose estimation of one of the robots influences not only at the low-level behaviour of this robot but also on the high-level task of the whole fleet. Our experience has shown us that most of the cases the high-level task has not been accurately carried out is because pose estimation was not properly solved.

It is usual to solve the pose estimation through image alignment when GPS or landmarks cannot be used. Nevertheless, depending on the environment, this is a really difficult task. For this reason, we have advocated for incorporating human interactivity, which only is demanded when the automatic image alignment solver is not able to achieve a good estimation. Our policy is based on it is worth to ask to a human to solve a low level task than not to achieve a high level task, although the human response could be slower than the automatic system. Nevertheless, note that while the human interacts on a pair of robots, the other ones can automatically solve the image alignment and pose estimation.

The novelty of the model presented in Chapter 7 is to use human interaction to improve the pose accuracy of a fleet of robots in a cooperative robotics framework. The system automatically aligns the 3D-points but manually aligns the 2D-points when it is needed. Both, the 2D and 3D cameras are embedded in the robots. In this chapter, we have only explored how the interactive image alignment is solved to achieve the pose estimation. We have depicted the whole framework and we have concretised on the modules that specifically deduces the pose of the fleet given the human interaction. The method we have commented is part of a larger project in which social robots guide people through outdoor or indoor scenes. Experimental section shows that it is clear that pose accuracy increases with only few human interactions. Moreover, the runtime of the alignment module (based on ICP algorithm) is reduced when the initial estimation is close to the solution. Therefore, an initial human interaction, not only makes the automatic system to achieve a better pose estimation but also to reduce the runtime of subsequent image alignments.

8.2 Future work and perspectives

The contributions of this thesis have been showed above. At this point it is necessary to think about what can be considered as future work and reflect on how will evolve the state of the art of the domain.

In Chapter 2 we have explored different definitions of local structures and how it is possible to define different metrics between them, it will be interesting as a future work to study how using cliques[5] of different order as a local structure (a local structure that can be seen as an extension of *Star*) affects the *Graph Edit Distance* performance in terms of classification accuracy and runtime.

Moreover, active and interactive strategies presented in Chapter 3, introduces a new strategy to find the "ideal" correspondence between two graphs after few interactions in which only is required to impose a partial mapping between nodes each iteration, but it have the limitation that not learns parameters during the iterative process. On contrary, in Chapter 4 and Chapter 5, it is presented a model for learn the parameters for *Error-Tolerant Graph Matching* but we need an amount of training examples (whole correspondences between graphs). As future work, the model for learning edit cost can be adapted to an online system to learn the weights at each oracle's interaction when it imposes only one node mapping using active queries strategies to choose the node that must be imposed. It also can be applied to multiple image registration or multiple graph correspondence problems incorporating consensus strategies.

In Chapter 6 it has been demonstrated that structured class prototypes continue to be usable and competitive although embedding methods are currently the most commonly used paradigm for graph classification. With respect to the evolution of the state of the art, always is difficult to predict how will evolve the domain, but given the excellent results of deep learning in other branches of pattern recognition and machine learning in general, it would not be surprising that the next paradigm for graph classification uses artificial neural networks. Schematically, converting graphs to vectors (a

[5] A subset of the nodes, $\Sigma_c \subseteq \Sigma_v$, such that every two distinct nodes are adjacent.

process similar to embedded methods) that can be processed as inputs by the input layers of neural networks.

Finally as a future work related to the cooperative framework presented in Chapter 7, since we consider the methodology has been validated in the *SAGRADA_FAMILIA* database, it will be interesting to integrate this method in a real-time robot environment.

Annex

Databases

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

I 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 [72] 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* (Chapter 3, Chapter 4 and Chapter 5). 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 ground-truth 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*.

I.1 General structure

Databases are composed of N registers of elements $\{G^{p^i}, G^{q^i}, \hat{f}^i, C^i\}$ that have a pairs of graphs and a correspondence. *attributed graphs* G^{p^i} and G^{q^i} 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^{q^i} may have some nodes of G^{p^i} mapped to nodes of G^{q^i}, and other ones mapped to the null node. Nevertheless, two nodes of G^{p^i} cannot be mapped to the same node of G^{q^i}. The null node is a mechanism to represent that a node of G^{p^i} do not have to be mapped to any node of G^{q^i} [35]. Note some nodes of G^{q^i} may not have been mapped to any node of G^{p^i} through f^i.

I.2 Databases

I.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.

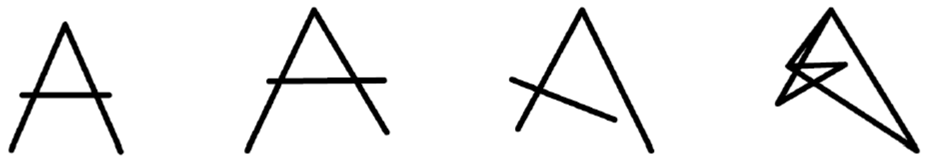


Figure A.1. Different instances of letter A.

I.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.

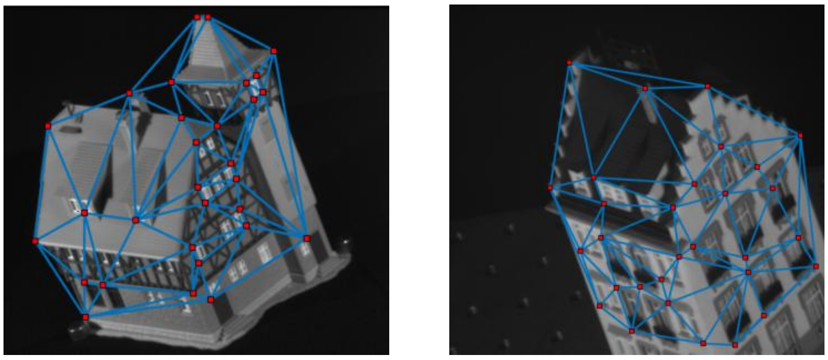

a. *HOUSE* b. *HOTEL*

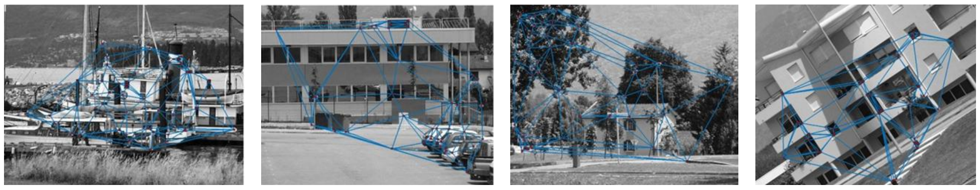
Figure A.2. Examples of HOUSE and HOTEL original images databases with salient points and edges.

I.2.c HORSE

Consists of a set of points from an original image (drawing taken from [148, 149, 150], with 35 hand-marked salient point). There are three datasets (*HORSE-NOISE*, *HORSE-SHEAR*, *HORSE-ROTATION*) artificially generated applying common transformations. Each node represents a salient point and contains 60 *Context Shape* features. Edges have been deduced through *Delaunay* triangulation. Due to distortions have been artificially created, the ground-truth correspondence have been easily deduced.

I.2.d 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 each sequence 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.

II Databases for graph classification

The following databases (part of the IAM graph repository [72]) are used in the experiments to evaluate the classification accuracy.

II.1 Databases

II.1.a GREC

The *GREC* database is composed of a set of 150 symbols from architecture, electronics and other technical fields. We have used a subset of 22 different symbols (classes), which are composed only of straight lines. These images are converted into graphs by assigning a node to each junction or terminal point and an edge to each line. For each of the 22 symbols in the dataset there are 50 distorted instances. So, totally we have 1100 images. The complete dataset is split into a reference set of 440 (20 graphs for each class) and a test set of 660 elements (30 graphs per class).

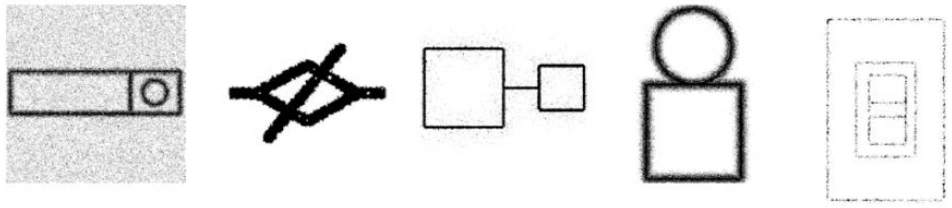


Figure A.4. Different original images with different distortion levels used to construct the instances for GREC database.

II.1.b COIL

The *COIL-RAG* database [214] consists of images of 100 different objects. Images of the objects are taken at pose intervals of 5 degrees. The images are segmented and transformed into graphs. Nodes feature the color histogram of the corresponding segment. Edges represent the adjacency regions featured with the length, in pixels, of the common border of two adjacent regions.

The *COIL-DEL* database uses the same original images as the *COIL-RAG* database but a different a different graph is generated for each image. Nodes represent 2D points detected using the *Harris Corners* detection [215], and edges the connections deduced using *Delaunay* triangulation.

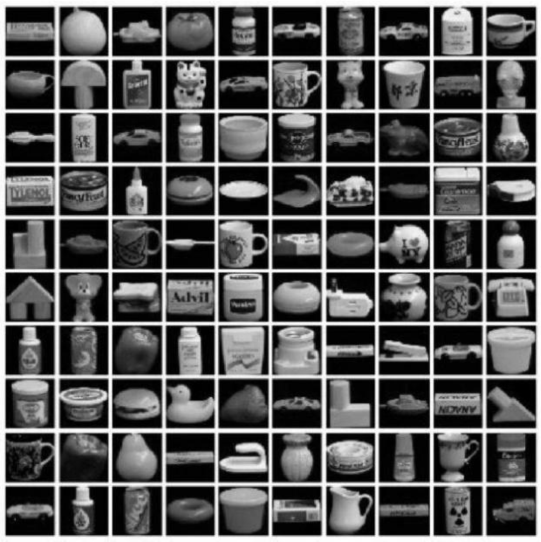


Figure A.5. Original images of COIL databases.

II.1.c FINGERPRINTS

The *FINGERPRINTS* database consists of a set of graphs generated from skeletonised 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-dimensional attribute giving its position. The edges are attributed with an angle denoting the orientation of the edge with respect to the horizontal direction.

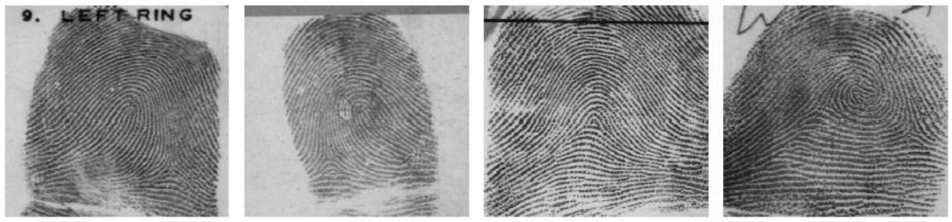


Figure A.6. Original examples of the four fingerprint classes.

II.1.d PROTEINS

The *PROTEINS* database consists of 600 graphs representing proteins originally used in [216]. It consists of six classes of proteins (100 per class), using reference, validation and test set of equal size (200). The proteins are converted into graphs. Nodes are labeled with their type and their amino acid sequence. Each node is connected with an edge to its three nearest neighbors in space. Edges are labeled with their type and the distance they represent in angstroms.

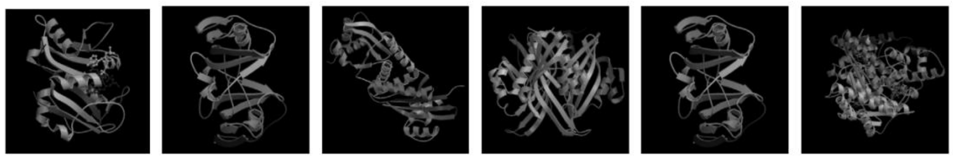


Figure A.7. Examples of 6 original images of the 6 different classes of proteins.

III "Sagrada Família" database

The *SAGRADA_FAMILIA* database was created using a sequence of 360 2D-pictures taken of "Sagrada Família" church in Barcelona (Spain). This sequence of pictures has been taken around the church and each picture is taken in an increase of approximate one degree with respect to the centre of the church. Given the whole sequence, we used the method called Bundler [212] to extract 100,532 3D-points of the church and the information of which 2D-pictures visualise these 3D points. Each picture has captured from 4,000 to 40,000 3D-points. Moreover, the method returns the relation between the 3D-points and the position in pixels in the pictures. Then, the positions of the cameras were deduced by the pose estimation method presented in [213].

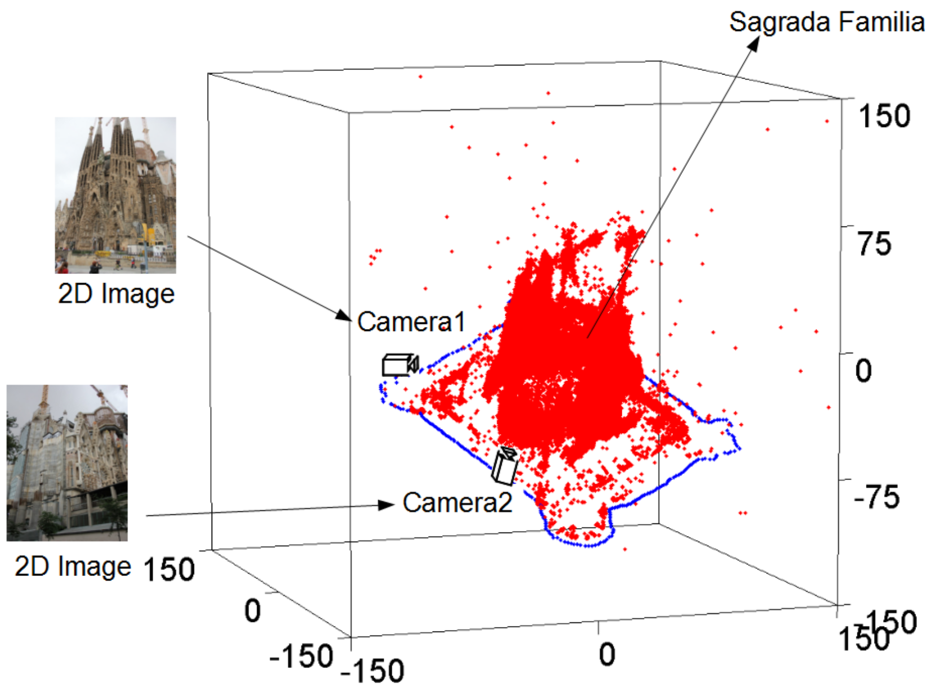
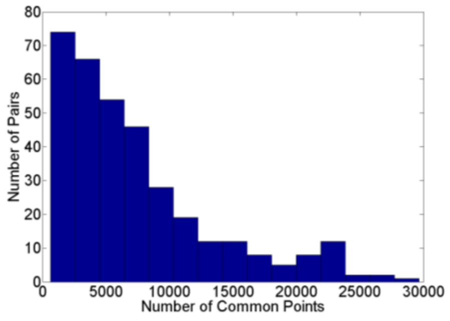


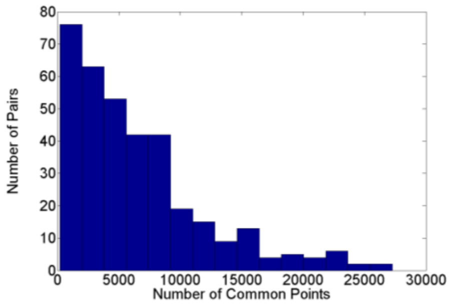
Figure A.8. "Sagrada Família" point cloud (red points) and the camera poses (blue points) in a 3D coordinate system (in meters) in which the origin of the axes is the centre of the church.

Figure A.8 shows the obtained 3D model of "Sagrada Família" (red points), and the different poses of the camera that has captured the images of the model (blue points). Axes are expressed in meters and the centre of the church is the origin of the coordinate system. Note there are some noisy points in the sky.

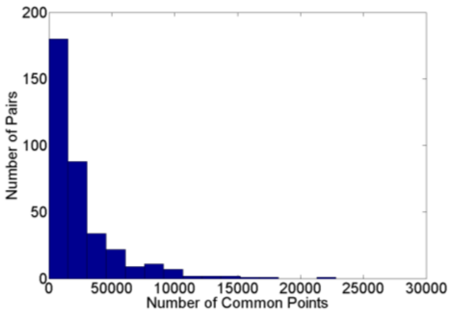
In general, when the number of 3D-points shared by two 3D-images increases, the alignment algorithms tend to find a better homography since the ratio of the signal versus noise tends to increase. We have paired each of the 360 images to another one that is T images far away in the sequence. Figure A.9 shows the number of pairs of pictures with respect to the number of 3D-points in common considering four different levels of separation between images (5, 20, 40, 60). Note the maximum separation is 180. There are 360 pairs but the area of each plot is slightly lower than 360 since there are few pairs of pictures with more than 30,000 points in common that have not been considered in the plot. Clearly, when the separation between pictures increases, the number of common points tends to reduce.



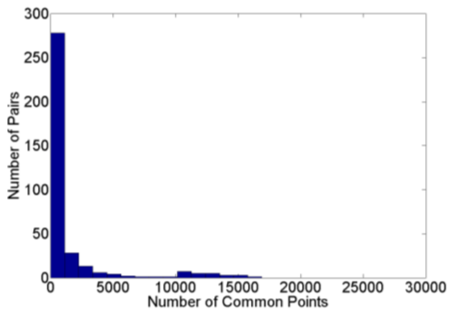
a. *Separation of 5 images*



b. *Separation of 20 images*



c. *Separation of 40 images*



d. *Separation of 60 images*

Figure A.9. Number of pairs of images with respect to the number of common 3D-points given different separation values.

List of Publications

Journals

1. Francesc Serratosa, Xavier Cortés, Albert Solé-Ribalta. "Component retrieval based on a database of graphs for Hand-Written Electronic-Scheme Digitalisation". Expert Syst. Appl. 40(7): 2493-2502, 2013.
2. Francesc Serratosa, Xavier Cortés. "Graph Edit Distance: Moving from global to local structure to solve the graph-matching problem". Pattern Recognition Letters 65: 204-210, 2015.
3. Xavier Cortés, Francesc Serratosa. "Learning graph-matching edit-costs based on the optimality of the oracle's node correspondences". Pattern Recognition Letters 56: 22-29, 2015.
4. Francesc Serratosa, Xavier Cortés. "Interactive graph-matching using active query strategies". Pattern Recognition 48(4): 1364-1373, 2015.
5. Xavier Cortés, Francesc Serratosa. "An interactive method for the image alignment problem based on partially supervised correspondence". Expert Syst. Appl. 42(1): 179-192, 2015.
6. Xavier Cortés, Francesc Serratosa. "Learning Graph Matching Substitution Weights Based on the Ground Truth Node Correspondence". IJPRAI 30(2), 2016.
7. Xavier Cortés, Francesc Serratosa. "Cooperative pose estimation of a fleet of robots based on interactive points alignment". Expert Syst. Appl. 45: 150-160, 2016.

Conferences

1. Francesc Serratosa, Albert Solé-Ribalta, Xavier Cortés. "Automatic Learning of Edit Costs Based on Interactive and Adaptive Graph Recognition". GbRPR 2011: 152-163, 2011.
2. Francesc Serratosa, Albert Solé-Ribalta, Xavier Cortés. "K-nn Queries in Graph Databases Using M-Trees". CAIP (1) 2011: 202-210, 2011.
3. Francesc Serratosa, Xavier Cortés, Albert Solé-Ribalta. "Graph Database Retrieval Based on Metric-Trees". SSPR/SPR 2012: 437-447, 2012.

4. Albert Solé-Ribalta, Xavier Cortés, Francesc Serratosa. “A Comparison between Structural and Embedding Methods for Graph Classification”. SSPR/SPR 2012: 234-242. 2012.
5. Xavier Cortés, Francesc Serratosa, Albert Solé-Ribalta. “Active Graph Matching Based on Pairwise Probabilities between Nodes”. SSPR/SPR 2012: 98-106, 2012.
6. Francesc Serratosa, Xavier Cortés, Albert Solé-Ribalta. “Interactive graph matching by means of imposing the pairwise costs”. ICPR 2012: 1298-1301, 2012.
7. Xavier Cortés, Francesc Serratosa. “Active-Learning Query Strategies Applied to Select a Graph Node Given a Graph Labelling”. GbRPR 2013: 61-70, 2013.
8. Xavier Cortés, Carlos Francisco Moreno-García, Francesc Serratosa. “Improving the Correspondence Establishment Based on Interactive Homography Estimation”. CAIP (2) 2013: 457-465, 2013.
9. Francesc Serratosa, Xavier Cortés. “Edit Distance Computed by Fast Bipartite Graph Matching”. S+SSPR 2014: 253-262, 2014.
10. Xavier Cortés, Francesc Serratosa. “Human interaction to improve the image alignment on a cooperative robotic framework”. ETFA 2014: 1-8, 2014.
11. Carlos Francisco Moreno-García, Xavier Cortés, Francesc Serratosa. “Partial to Full Image Registration Based on Candidate Positions and Multiple Correspondences”. CIARP 2014: 745-753, 2014.
12. Xavier Cortés, Carlos Francisco Moreno-García, Francesc Serratosa, “Learning Graph-Matching Substitution Costs Based on the Optimality of the Oracle's Correspondence”. CIARP 2014: 506-514, 2014.
13. Xavier Cortés, Francesc Serratosa, Carlos Francisco Moreno-García. “An Interactive Model for Structural Pattern Recognition based on the Bayes Classifier”. ICPRAM (1) 2015: 240-247, 2015.
14. Carlos Francisco Moreno-García, Xavier Cortés, Francesc Serratosa. “Iterative Versus Voting Method to Reach Consensus Given Multiple Correspondences of Two Sets”. IbPRIA 2015: 530-540, 2015.
15. Xavier Cortés, Francesc Serratosa, Carlos Francisco Moreno-García. “On the Influence of Node Centralities on Graph Edit Distance for Graph Classification”. GbRPR 2015: 231-241, 2015.
16. Carlos Francisco Moreno-García, Francesc Serratosa, Xavier Cortés. “Consensus of Two Graph Correspondences Through a Generalisation of the Bipartite Graph Matching”. GbRPR 2015: 87-97, 2015.

17. Xavier Cortés, Francesc Serratosa, Carlos Francisco Moreno-García. "Ground Truth Correspondence Between Nodes to Learn Graph-Matching Edit-Costs". CAIP (1) 2015: 113-124, 2015.

Workshops

1. Xavier Cortés. "Learning Edit Costs for Graph-Matching". 1st URV Doctoral Workshop in Computer Science and Mathematics, 77, 2014.

Bibliography

1. K. Driessens, J. Ramon, T. Gärtner. "Graph kernels and Gaussian processes for relational reinforcement learning". Machine Learning 64(1-3): 91-119, 2006
2. P. Mahé, J.-P. Vert. "Graph kernels based on tree patterns for molecules". Machine Learning 75(1):3-35, 2009.
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". International Journal of Pattern Recognition and Artificial Intelligence, Vol. 18, No. 3 pp: 265-298, 2004.
6. B. Luo, R.Wilson, E. Hancock. "Spectral feature vectors for graph clustering". Structural, Syntactic, and Statistical Pattern Recognition, pages 83–93, 2002.
7. A. Robles-Kelly, E.R. Hancock. "Graph edit distance from spectral seriation". IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(3):365–378, 2005.
8. S. Kosinov, T. Caelli. "Inexact multisubgraph matching using graph eigenspace and clustering models". Structural, Syntactic, and Statistical Pattern Recognition, pages 133–142, 2002.
9. R.C. Wilson, E.R. Hancock, and B. Luo. "Pattern vectors from algebraic graph theory". IEEE Trans. on Pattern Analysis and Machine Intelligence, 27(7):1112–1124, 2005.
10. H. Qiu, E.R. Hancock. "Graph matching and clustering using spectral partitions". Pattern Recognition, 39(1):22–34, 2006.
11. J. Ramon, T. Gärtner. "Expressivity versus efficiency of graph kernels". First International Workshop on Mining Graphs, Trees and Sequences, pages 65–74, 2003.
12. 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.
13. S.V.N Vishwanathan, K. Borgwardt, N.N. Schraudolph. "Fast computation of graph kernels". pp: 1449-1456, Annual Conference on Neural Information Processing Systems. MIT Press, 2006.
14. B. Gauzere, L. Brun, D. Villemin. "Two new graph kernels and applications to chemoinformatics". Int. Workshop on Graph Based Representations in Pattern Recognition, pages 112–121, 2011.

15. B. Gauzere, P.A. Grenier, L. Brun, D. Villemin. “Treelet kernel incorporating cyclic, stereo and inter pattern information in chemoinformatics”. Pattern Recognition, 48(2):356–367, 2015.
16. 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.
17. P. Foggia, G. Percannella, M. Vento. “Graph matching and learning in pattern recognition in the last 10 years”. Int. Journal of Pattern Recognition and Art. Intelligence, 28(1) 2014.
18. A. Sanfeliu, K. Fu. “A distance measure between attributed relational graphs for pattern recognition”. IEEE Trans. on Sys., Man and Cybern., 13, pp. 353-362, 1983.
19. H. Bunke, G. Allermann. “Inexact graph matching for structural pattern recognition”. Pattern Recognition Letters, 1(4): p. 245-253, 1983.
20. H. Bunke. “Handbook of Fingerprint Recognition common subgraph”. Pattern Recognition Letters, 1998. 18(8): p. 689-694.
21. H. Bunke. “Error Correcting Graph Matching: On the Influence of the Underlying Cost Function”. Transactions on Pattern Analysis and Machine Intelligence, 21(9): p. 917-922, 1999.
22. A.K.C. Wong, M. You. “Entropy and distance of random graphs with application to structural pattern recognition”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 7, pp. 599-609, 1985.
23. F. Serratosa, R. Alquézar, A. Sanfeliu. “Function-described graphs for modeling objects represented by attributed graphs”. Pattern Recognition, 36 no. 3, pp. 781-798, 2003.
24. A. Sanfeliu, F. Serratosa, R. Alquézar. “Second-Order Random Graphs for modeling sets of Attributed Graphs and their application to object learning and recognition”. Intern. Journal of Pattern Recognition and Artificial Intelligence, 18, no. 3, pp. 375-396, 2004.
25. K. Riesen, H. Bunke. “Approximate graph edit distance computation by means of bipartite graph matching”. Image and Vision Computing, vol. 27, no. 4, pp. 950–959, 2009.
26. R. Burkard, M. Dell’Amico, and S. Martello, “Assignment Problems”. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.
27. A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, H. Bunke. “Approximation of graph edit distance based on Hausdorff matching”. Pattern Recognition 48(2): 331-343, 2015.
28. D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge. “Comparing images using the Hausdorff distance”. IEEE Trans. Pattern Anal. Mach. Intell. 15, pp: 850–863, 1993.
29. F. Serratosa. “Speeding up Fast Bipartite Graph Matching trough a new cost matrix”. International Journal of Pattern Recognition and Artificial Intelligence, 29 (2), 2015.

30. G. Fekete, J.O. Eklundh, A. Rosenfeld. "Probabilistic Relaxation: Evaluation and Applications". IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 3, No. 4, 459-469, 1981.
31. R. Jonker, T. Volgenant. "A shortest augmenting path algorithm for dense and sparse linear assignment problems". Computing 38 pp: 325-340, 1987.
32. S. Gold, A. Rangarajan. "A Graduated Assignment Algorithm for Graph Matching". IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 18 No. 4, 377 – 388, 1996.
33. L. Bin, E.R. Hancock. "Structural graph matching using the EM algorithm and singular value decomposition" IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 23, No. 10, 1120-1136, 2001.
34. J.A. Snyman. "Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms". Springer Publishing. ISBN 0-387-24348-8, 2005.
35. 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.
36. H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, K. Tsuda. "gBoost: a mathematical programming approach to graph classification and regression". Machine Learning 75(1): 69-89, 2009.
37. M. Neuhaus, H. Bunke. "Self-organizing maps for learning the edit costs in graph matching". IEEE Trans. on Sys., Man, and Cybernetics, Part B 35(3), pp. 503-514, 2005.
38. M. Neuhaus, H. Bunke. "Automatic learning of cost functions for graph edit distance". Inf. Sci. 177(1), pp. 239-247, 2007.
39. T.S. Caetano, J.J. McAuley, L. Cheng, Q.V. Le, A.J. Smola. "Learning Graph Matching". IEEE Trans. Pattern Anal. Mach. Intell. 31(6): 1048-1058, 2009.
40. M. Leordeanu, R. Sukthankar, M. Hebert. "Unsupervised Learning for Graph Matching". International Journal of Computer Vision 96(1): 28-45, 2012.
41. K. Teuvo. "Self-Organized Formation of Topologically Correct Feature Maps". Biological Cybernetics 43 (1): 59–69, 1982.
42. A.P. Dempster, N.M. Laird, D.B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". Journal of Royal Statist. Society, Series B 39(1):1–38, 1977.
43. K. Anstreicher. "Recent Advances in the Solution of Quadratic Assignment Problems". Math. Programming, Ser. B, vol. 97, pp. 27-42, 2003.
44. E. Alpaydin. "Introduction to machine learning". Cambridge, Mass. [u.a.]: MIT Press. pp. 79, 80, 2011.
45. M. Leordeanu, M. Hebert. "A Spectral Technique for Correspondence Problems Using Pairwise Constraints". ICCV 2005: 1482-1489, 2005.
46. K. Konstantinidis, A. Gasteratos, I. Andreadis. "Image retrieval based on fuzzy colour histogram processing". Optics Communications, 248, pp: 375–386, 2005.

47. R. Datta, D. Joshi, J. Li, J.Z. Wang. "Image Retrieval: Ideas, Influences, and Trends of the New Age". ACM Computing Surveys, 40 (2), Article 5, 2008.
48. J. Salim, T. Salvatore. "Hypergraph-based image retrieval for graph-based representation". Pattern Recognition, 45 (11), pp: 4054-4068, 2012.
49. J. Lebrun, P. Gosselin, S. Philipp. "Inexact graph matching based on kernels for object retrieval in image databases". Image and Vision Computing, 29 (11), pp: 716-729, 2011.
50. B. Le Saux, H. Bunke. "Feature selection for graph-based image classifiers". IbPRIA 2005. LNCS, vol. 3523, pp. 147–154, 2005.
51. M. Neuhaus, K. Riesen, H. Bunke. "Fast Suboptimal Algorithms for the Computation of Graph Edit Distance". Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition, LNCS 4109, pp. 163-172, 2006.
52. X. Yan, P.S. Yu, J. Han. "Graph indexing: a frequent structure-based approach". ACM SIGMOD international conference on Management of data, pp. 335-346, 2004.
53. D. Shasha, J.T.L. Wang, R. Giugno. "Algorithmics and applications of tree and graph searching". ACM SIGMOD-SIGACT-SIGART, pp. 39-52, 2002.
54. S. Berretti, A. Del Bimbo, E. Vicario. "Efficient Matching and Indexing of Graph Models in Content-Based Retrieval". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 10, pp. 1089-1105, 2001.
55. H. He, A.K. Singh. "Closure-Tree: An Index Structure for Graph Queries". Proc. International Conference on Data Engineering, pp. 38, 2006.
56. S.Y. Lee, F. Hsu. "Spatial Reasoning and Similarity Retrieval of Images using 2D C-Strings Knowledge Representation". Pattern Recognition, 25 no. 3, pp. 305-318, 1992.
57. P. Ciaccia, M. Patella, P. Zezula. "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces". Proc. 23rd VLDB Conference, pp. 426-435, 1997.
58. M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, H. Bunke. "Generalized Median Graph Computation by Means of Graph Embedding in Vector Spaces". Pattern Recognition, 43 (4), pp. 1642-1655, 2010.
59. X. Jiang, A. Münger, H. Bunke. "On median graphs: Properties, algorithms and applications". IEEE Trans. on PAMI, vol. 23, no. 10, pp: 1144-1151, 2001.
60. X. Jiang, A. Münger, H. Bunke. "On median graphs: Properties, algorithms and applications". IEEE Trans. on PAMI, vol. 23, no. 10, pp: 1144-1151, 2001.
61. M. Ferrer, E. Valveny, F. Serratosa. "Median graphs: A genetic approach based on new theoretical properties". Pattern Recognition, vol. 42, no. 9, pp. 2003-2012, 2009.
62. M. Ferrer, E. Valveny, F. Serratosa. "Median graph: A new exact algorithm using a distance based on the maximum common subgraph". Pattern Recognition Letters 30(5), pp: 579-588, 2009.

63. F. Serratosa, A. Solé, E. Vidiella. “Graph Indexing and Retrieval based on Median Graphs”. Mexican Conference on Pattern Recognition, LNCS 6256, pp:311-321, 2010.
64. H. Bunke, A. Munger, X. Jiang. “Combinatorial search versus genetic algorithms: A case study based on the generalized median graph problem”. Pattern Recognition Letter, 20, pp: 1271-1277, 1999.
65. A. Solé, F. Serratosa. “Graduated Assignment Algorithm for Finding the Common Labelling of a set of Graphs”. Syntactic and Structural Pattern Recognition, SSPR2010, Izmir, Turkey, LNCS 6218, pp: 180-190, 2010.
66. F. Serratosa, R. Alquézar, A. Sanfeliu. “Synthesis of Function-Described Graphs and clustering of Attributed Graphs”. International Journal of Pattern Recognition and Artificial Intelligence, vol. 16, no. 6, pp. 621-655, 2002.
67. F. Serratosa. “Fast Computation of Bipartite Graph Matching”. Pattern Recognition Letters 45, pp: 244 - 250, 2014.
68. K. Riesen, M. Ferrer, A. Fischer, H. Bunke. “Approximation of Graph Edit Distance in Quadratic Time”. Graph Based Representations in Pattern Recognition, pp: 3-12, 2015.
69. H. W. Kuhn. “The Hungarian Method for the assignment problem”. Naval Research Logistics Quarterly, 2: 83–97, 1955.
70. G. Peris, A. Marzal. “Fast Cyclic Edit Distance Computation with Weighted Edit Costs in Classification”. International Conference on Pattern Recognition, vol 4. pp: 184-187 2002.
71. L. Page, S. Brin, M. Rajeev, W. Terry. “The PageRank Citation Ranking: Bringing Order to the Web”. Technical Report. Stanford InfoLab, 1999.
72. K. Riesen, H. Bunke. “IAM graph database repository for graph based pattern recognition and machine learning”. in Structural, Syntactic, and Statistical Pattern Recognition, ser. LNCS 5342, N. da Vitoria Lobo et al., Ed., pp. 287–297, 2008.
73. T. Mitchell. “Machine Learning”. McGraw-Hill International Edit, ISBN-13: 978-0071154673, 1997.
74. A.D. Sappa, J. Vitrià, “Multimodal Interaction in Image and Video Applications”. Book Series Intelligent Systems Reference Library, 48, Springer, 2013.
75. F. Toselli, *et. al*. “Multimodal interactive pattern recognition and applications”. Springer, 2011.
76. A. Sanchís, A. Juan, E. Vidal. “A Word-Based Naïve Bayes Classifier for Confidence Estimation in Speech Recognition”. IEEE Transactions on Audio, Speech & Language Processing 20(2): 565-574, 2012.
77. C. Harris, M. Stephens. “A combined corner and edge detector”. Proceedings of the 4th Alvey Vision Conference. pp. 147–151, 1988.
78. D.G. Lowe. “Object recognition from local scale-invariant features”. Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157, 1999.

79. G. Sanromà, R. Alquézar, F. Serratosa, B. Herrera. "Smooth Point-set Registration using Neighbouring Constraints". Pattern Recognition Letters, 33, pp: 2029-2037, 2012.
80. G. Sanromà, R. Alquézar, F. Serratosa. "A New Graph Matching Method for Point-Set Correspondence using the EM Algorithm and Softassign". Computer Vision and Image Understanding, 116(2), pp: 292-304, 2012.
81. T.S. Caetano, T. Caelli, D. Schuurmans, D.A.C. Barone. "Graphical Models and Point Pattern Matching". IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 28, no. 10, pp. 1646-1663, 2006.
82. N. Rebagliati, A. Solé, M. Pelillo, F. Serratosa. "Computing the Graph Edit Distance Using Dominant Sets". International Conference on Pattern Recognition, ICPR2012, Tsukuba, Japan, pp: 1080-1083, 2012.
83. K. Mikolajczyk, C. Schmid. "A performance evaluation of local descriptors". IEEE Transactions on Pattern Analysis & Machine Intelligence, 27(10): 1615–1630, 2005.
84. B. Settles, "Active Learning Literature Survey". Computer Science Technical Report 1648, University of Wisconsin-Madison.
85. Q. Zhang, S. Sun. "Multiple-view multiple-learner active learning". Pattern Recognition, 43(9), pp: 3113-3119, 2010.
86. S. B. Kotsiantis. "Supervised Machine Learning: A Review of Classification Techniques". Informatica 31, pp: 249-268, 2007.
87. D. Yu, B. Varadarajan, L. Deng, A. Acero. "Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion". Computer Speech & Language, 24(3), pp: 433-444, 2010.
88. D. Zhang, F. Wang, Z. Shi, C. Zhang. "Interactive localized content based image retrieval with multiple-instance active learning". Pattern Recognition, 43(2), pp: 478-484, 2010.
89. M. Cord, P.H. Gosselin, S. Philipp-Foliguet. "Stochastic exploration and active learning for image retrieval" Image and Vision Computing, 25(1) pp: 14-23, 2007.
90. J. Cheng, K. Wang. "Active learning for image retrieval with Co-SVM". Pattern Recognition, 40(1), pp: 330-334, 2007.
91. P.H. Gosselin, F. Precioso, S. Philipp-Foliguet. "Incremental kernel learning for active image retrieval without global dictionaries". Pattern Recognition, 44(10–11), pp: 2244-2254, 2011.
92. A. Baranes, P.Y. Oudeyer. "Active learning of inverse models with intrinsically motivated goal exploration in robots". Robotics and Autonomous Systems, Available online, 2012.
93. A.H. Toselli, V. Romero, M. Pastor, E. Vidal. "Multimodal interactive transcription of text images". Pattern Recognition, 43(5), pp: 1814-1825, 2010.
94. V. Romero, A.H. Toselli, E. Vidal. "Multimodal interactive handwritten text transcription". Word Scientific, 2012.

95. R. Wang, S. Kwong, D. Chen. "Inconsistency-based active learning for support vector machines". Pattern Recognition, 45(10), Pages 3751-3767, 2012.
96. D. Gorisse, M. Cord, F. Precioso. "SALSAS: Sub-linear active learning strategy with approximate k-NN search". Pattern Recognition, 44 (10–11), pp: 2343-2357, 2011.
97. S. Patra, L. Bruzzone. "A cluster-assumption based batch mode active learning technique". Pattern Recognition Letters, 33(9), pp: 1042-1048, 2012.
98. E. Lughofer. "Hybrid active learning for reducing the annotation effort of operators in classification systems". Pattern Recognition, 45(2), pp: 884-896, 2012.
99. A. Tavares da Silva, A.X. Falcão, L.P. Magalhães. "Active learning paradigms for CBIR systems based on optimum-path forest classification". Pattern Recognition, 44(12) pp: 2971-2978, 2011.
100. Z. Wang, S. Yan, C. Zhang. "Active learning with adaptive regularization". Pattern Recognition, 44(10–11), pp: 2375-2383, 2011.
101. D. Angluin. "Queries and concept learning". Machine Learning, 2, pp:319–342, 1988.
102. D. Angluin. "Queries revisited". International Conference on Algorithmic Learning Theory, pp: 12–31, 2001.
103. D. Cohn, Z. Ghahramani, M.I. Jordan. "Active learning with statistical models". Journal of Artificial Intelligence Research, 4, pp: 129–145, 1996.
104. K. Lang, E. Baum. "Query learning can work poorly when a human Oracle is used". IEEE International Joint Conference on Neural Networks, pp: 335–340, 1992.
105. H. Yu. "SVM selective sampling for ranking with application to data retrieval". Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD), pp: 354–363. ACM Press, 2005.
106. D. Cohn, L. Atlas, R. Ladner. "Improving generalization with active learning". Machine Learning, 15(2), pp: 201–221, 1994.
107. S. Dasgupta, D.J. Hsu. "Hierarchical sampling for active learning". International Conference on Machine Learning, pp: 208–215, 2008.
108. I. Dagan, S. Engelson. "Committee-based sampling for training probabilistic classifiers". International Conference on Machine Learning, pp: 150–157, 1995.
109. T. Kanamori. "Pool-based active learning with optimal sampling distribution and its information geometrical interpretation". Neurocomputing, 71(1–3), pp: 353-362, (2007).
110. D. Lewis, W. Gale. "A sequential algorithm for training text classifiers". Conference on Research and Development in Information Retrieval, pp: 3–12, 1994.
111. B. Settles, M. Craven. "An analysis of active learning strategies for sequence labelling tasks". Conference on Empirical Methods in Natural Language Processing, pp: 1069–1078, 2008.

112. A. Culotta, A. McCallum. “Reducing labelling effort for structured prediction tasks”. National Conference on Artificial Intelligence, pp: 746–751, 2005.
113. R. Hwa. “Sample selection for statistical parsing”. Computational Linguistics, 30 (3), pp:73–77, 2004.
114. C.E. Shannon. “A mathematical theory of communication”. Bell System Technical Journal, 27, pp: 379–423,623–656, 1948.
115. H.S. Seung, M. Opper, H. Sompolinsky. “Query by committee”. ACM Workshop on Computational Learning Theory, pp: 287–294, 1992.
116. A. McCallum, K. Nigam. “Employing EM in pool-based active learning for text classification”. International Conference on Machine Learning, pp: 359–367, 1998.
117. P. Melville, R. Mooney. “Diverse ensembles for active learning”. International Conference on Machine Learning, pp: 584–591, 2004.
118. B. Settles, M. Craven, S. Ray. “Multiple-instance active learning”. Advances in Neural Information Processing Systems, 20, pp: 1289–1296, 2008.
119. T. Zhang, F.J. Oles. “A probability analysis on the value of unlabelled data for classification problems”. International Conference on Machine Learning, pp: 1191–1198, 2000.
120. N. Roy, A. McCallum. “Toward optimal active learning through sampling estimation of error reduction”. International Conference on Machine Learning, pp: 441–448, 2001.
121. X. Zhu, J. Lafferty, Z. Ghahramani. “Combining active learning and semisupervised learning using Gaussian fields and harmonic functions”. Workshop on the Continuum from Labelled to Unlabelled Data, pp: 58–65, 2003.
122. Y. Guo, R. Greiner. “Optimistic active learning using mutual information”. Proceedings of International Joint Conference on Artificial Intelligence, pp: 823–829, 2007.
123. F. Serratosa, X. Cortés, A. Solé-Ribalta. “Component Retrieval based on a Database of Graphs for Hand-Written Electronic-Scheme Digitalisation”. Expert Systems With Applications, (40), pp: 2493-2502, 2013.
124. S. Chakraborty, V. Balasubramanian, S. Panchanathan. “Generalized batch mode active learning for face-based biometric recognition”. Pattern Recognition, 46, (2), pp: 497-508, 2013.
125. D. Tuia, J. Muñoz-Marí, G. Camps-Valls. “Remote sensing image segmentation by active queries”. Pattern Recognition, 45 (6), pp: 2180-2192, 2012.
126. V. Vu, N. Labroche, B. Bouchon-Meunier. “Improving constrained clustering with active query selection”. Pattern Recognition, 45 (4), pp: 1749-1758, 2012
127. O. Chapelle, B. Schölkopf, A, Zien. “Semi-supervised learning”. Cambridge, Mass.: MIT Press, 2006.
128. O. Chapelle, B. Schölkopf, A, Zien. “Semi-supervised learning”. Cambridge, Mass.: MIT Press, 2006.

129. M. Leordeanu, M. Hebert, R. Sukthankar. "An integer projected fixed point method for graph matching and MAP inference". Neural Information Processing Systems, 2009.
130. B. Xiao, S. Yi-Zhe, P. Hall. "Learning invariant structure for object identification by using graph methods". Computer Vision and Image Understanding 115, pp: 1023–1031, 2011.
131. A. Sanfeliu, F. Serratosa, R. Alquézar. "Second-Order Random Graphs for modelling sets of Attributed Graphs and their application to object learning and recognition". International Journal of Pattern Recognition and Artificial Intelligence,18 (3) pp: 375-396, 2004.
132. F. Serratosa, R. Alquézar, A. Sanfeliu. "Function-Described Graphs for modelling objects represented by attributed graphs". Pattern Recognition 36 (3), pp. 781-798, 2003.
133. R. Raveaux, S. Adam, P. Héroux, E. Trupin. "Learning graph prototypes for shape recognition". Computer Vision and Image Understanding 115, pp: 905–918, 2011.
134. C.-R. Chou, B. Frederick, G. Mageras, S. Chang, S. Pizer. "2D/3D image registration using regression learning". Computer Vision and Image Understanding 117, pp: 1095–1106, 2013.
135. E.Z. Borzeshi, M. Piccardi, K. Riesen, H. Bunke. "Discriminative prototype selection methods for graph embedding". Pattern Recognition 46(6): 1648-1657, 2013.
136. H. Bunke, K. Riesen. "Towards the unification of structural and statistical pattern recognition". Pattern Recognition Letters 33(7): 811-825, 2012.
137. J. Hou, M. Pelillo. "A simple feature combination method based on dominant sets". Pattern Recognition 46, pp: 3129-3139, 2013.
138. H. Lei, K. Mei, N. Zheng, P. Dong, N. Zhou, J. Fan. "Learning group-based dictionaries for discriminative image representation". Pattern Recognition, 47(2): 899–913, 2014.
139. J. Richarz, S. Vajda, R. Grzeszick, A. Gernot. "Fink Semi-supervised learning for character recognition in historical archive documents". Pattern Recognition, 2013.
140. N. Hu, R.M. Rustamov, L.J. Guibas. "Graph Matching with Anchor Nodes: A Learning Approach". CVPR, 2906-2913, 2013.
141. I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun. "Large Margin Methods for Structured and Interdependent Output Variables". Journal Machine Learning Research, vol. 6, pp. 1453-1484, 2005.
142. R.H. Byrd, R.B. Schnabel, G.A. Schultz. "A trust region algorithm for nonlinearly constrained optimization". SIAM J. Numer. Anal., 24, pp. 1152–1170, 1987.
143. C. T. Kelley. "Iterative Methods for Optimization". SIAM Frontiers in Applied Mathematics, no 18, ISBN 0-89871-433-8, 1999.
144. K.I.M McKinnon. "Convergence of the Nelder–Mead simplex method to a non-stationary point". SIAM J Optimization 9: 148–158, 1999.

145. C. Teo, Q. Le, A. Smola, S. Vishwanathan. "A Scalable Modular Convex Solver for Regularized Risk Minimization". Proc. Knowledge Discovery and Data Mining, 2007.
146. http://deim.urv.cat/~francesc.serratosa/SW/
147. http://vasc.ri.cmu.edu/idb/html/motion/
148. A.M. Bronstein, M.M. Bronstein, R. Kimmel. "Numerical Geometry of Non-Rigid Shapes". Springer, 2007.
149. A.M. Bronstein, M.M. Bronstein, A.M. Bruckstein, R. Kimmel. "Analysis of Two-Dimensional Non-Rigid Shapes". International Journal Computer Vision, 2007.
150. Mythological Creatures 2D database, http://tosca.cs.technion.ac.il, 2009.
151. F. Serratosa, A. Solé, X. Cortés. "Automatic Learning of Edit Costs based on Interactive & Adaptive Graph Recognition". Graph based Representations, GbR2011, Munster, Germany, LNCS 6658 pp: 152,163, 2011.
152. J.A. Nelder, R. Mead. Computer Journal, vol 7, pp 308-313, 1965.
153. J.L. Zhao, H.K. Cheng. "Graph Indexing for Spatial Data Traversal in Road Map Databases". Computers & Operations Research, vol. 28, pp: 223-241, 2001.
154. A. Fernández, S. Gómez. "Solving Non-uniqueness in Agglomerative Hierarchical Clustering Using Multidendrograms". Journal of Classification (25), pp: 43–65, 2008.
155. M.A. Lozano, F. Escolano, B. Bonev, P. Suau, P., W. Aguilar, J.A. Saez, M.A. Cazorla, "Region and constellations based categorization of images with unsupervised graph learning". Image and Vision Computing 27 pp: 960–978, 2009.
156. A. Solé, F. Serratosa. "Graduated Models and Algorithms for computing the Common Labelling of a set of Attributed Graphs". CVIU, 115 (7), pp: 929-945, 2011.
157. F. Serratosa, A. Solé, "A Probabilistic Framework to obtain a Common Labelling between Attributed Graphs". Iberian Conference on Pattern Recognition and Image Analysis, IBPRIA2011, Gran Canaria, Spain, LNCS 6669, pp: 180-190, 2011.
158. K. Riesen, H. Bunke. "Graph classification based on vector space embedding". International Journal of Pattern Recognition and Artificial Intelligence 23(6), pp: 1053–1081, 2009.
159. H. Bunke, K. Riesen. "Towards the unification of structural and statistical pattern recognition". Pattern Recognition Letters 33(7), 811–825, 2012.
160. C. Trevai, Y. Fukazawa, J. Ota, H. Yuasa, T. Arai, H. Asama. "Cooperative exploration of mobile robots using reaction-diffusion equation on a graph". IEEE International Conference on Robotics and Automation, vol. 2, 2003.
161. J. Casper, RR. Murphy. "Human–robot interactions during the robot-assisted urban search and rescue response at the world trade centre". IEEE Transactions on Systems, Man, and Cybernetics, Part B. 33: 367–385, 2003.

162. S.-G. Kim, S. Taguchi, S.-I. Hong, H.-H. Lee. "Cooperative behavior control of robot group using stress antibody allotment reward". Artificial life and robotics 19 (1), pp: 16-22, 2014.
163. A. Garrell, A. Sanfeliu. "Cooperative social robots to accompany groups of people". International Journal of Robotics Research, 31(13): 1675-1701, 2012.
164. B. Zitová, J. Flusser. "Image registration methods: a survey". Image Vision Comput. 21(11): 977-1000, 2003.
165. J. Salvi, C. Matabosch, D. Fofi, J. Forest. "A review of recent range image registration methods with accuracy evaluation". Image Vision Comput. 25(5): 578-596, 2007.
166. F. Serratosa, R. Alquézar, N. Amézquita. "A Probabilistic Integrated Object Recognition and Tracking Framework". Expert Systems With Applications, 39, pp: 7302-7318, 2012.
167. http://www.optitrack.com/
168. http://www.iri.upc.edu/project/show/144
169. I. Huerta, M. Pedersoli, J. Gonzàlez, A. Sanfeliu. "Combining where and what in change detection for unsupervised foreground learning in surveillance". Pattern Recognition, 48(3): 709-719, 2015.
170. X. Cortés, F. Serratosa. "An Interactive Method for the Image Alignment problem based on Partially Supervised Correspondence". Expert Systems With Applications 42 (1), pp: 179 - 192, 2015.
171. F. Serratosa, X. Cortés. "Interactive Graph-Matching using Active Query Strategies". Pattern Recognition 48, pp: 1360-1369, 2015.
172. J.-P. Thirion. "New feature points based on geometric invariants for 3D image registration". International Journal of Computer Vision 18 (2), pp: 121-137, 1996.
173. X. Cortés, Moreno, F. Serratosa. "Improving the Correspondence Establishment based on Interactive Homography Estimation". Computer Analysis of Images and Patterns, CAIP2013, York, Unated Kindom, LNCS 8048 , pp: 457-465, 2013.
174. M. Villamizar, J. Andrade-Cetto, A. Sanfeliu, F. Moreno-Noguer. "Bootstrapping Boosted Random Ferns for discriminative and efficient object classification". Pattern Recognition 45(9) pp: 3141-3153, 2012.
175. M. Xu, M. Petrou. "3D Scene interpretation by combining probability theory and logic: The tower of knowledge". Computer Vision and Image Understanding, 115 (11), pp. 1581-1596, 2011.
176. C. Tomasi, T. Kanade. "Detection and tracking of point features". Tech. Rep. CMUCS-91-132, Carnegie Mellon University, 1991.
177. W. Han, M. Brady. "Real-time corner detection algorithm for motion estimation". Image and Vision Computing, pp: 695–703, 2005.
178. E. Rosten, T. Drummond. "Machine learning for high-speed corner detection". European Conference on Computer Vision, vol. 1, pp. 430–443, 2006.

179. Z. Zhang. "Iterative point matching for registration of free-form curves and surfaces". Int. J. Comput. Vision 13 (2), pp: 119–152, 1994.
180. M.A. Fischler, R.C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". Commun. ACM 24 (6), pp: 381–395, 1981.
181. B. Lia, *et. al*. "A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries". Computer Vision and Image Understanding 131, pp: 1–27, 2015.
182. B. Luo, E. Hancock. "A unified framework for alignment and correspondence". Computer Vision and Image Understanding 92 (1), 26–55, 2003.
183. A. Rangarajan, H. Chui, F.L. Bookstein. "The softassign procrustes matching algorithm". Proceedings of the 15th International Conference on Information Processing in Medical Imaging. Springer-Verlag, pp. 29–42,1997.
184. A. Myronenko, X.B. Song. "Point set registration: coherent point drift". IEEE Trans. Pattern Anal. Machine Intell. 32 (12), 2262–2275, 2010.
185. H. Chui, A. Rangarajan. "A new point matching algorithm for non-rigid registration". Computer Vision and Image Understanding 89 (2–3), 114–141, 2003.
186. T. Pfluger, *et al*. "Quantitative Comparison of Automatic and Interactive Methods for MRI-SPECT Image Registration of the Brain Based on 3-Dimensional Calculation of Error". Journal of Nuclear Medicine 41(11), 1823-1829, 2000.
187. U. Pietrzyk, *et al*. "An Interactive Technique for Three-Dimensional Image Registration". Validation for The Journal of Nuclear Medicine, 35(12), 2011-2018, 1994.
188. M. Khader, A. Ben Hamza. "An information-theoretic method for multimodality medical image registration". Expert Systems with Applications 39 (5), 5548-5556, 2012.
189. B.J. Von, U. Neitzel. "Interactive image registration". WO 2010134013 A1, 2010.
190. Gering, David T. "Systems and methods for interactive image registration". U.S. Patent No. 7,693,349, 6 Apr. 2010.
191. S. Bianco, G. Ciocca, P. Napoletano, R. Schettini. "An interactive tool for manual, semi-automatic and automatic video annotation". Computer Vision and Image Understanding 131, pp: 88–99, 2015.
192. http://www.featurespace.org.
193. F. Schwenker, E. Trentin. "Pattern Classification and Clustering: a Review of Partially Supervised Learning Approaches". Pattern Recognition Letters, Volume 37, Pages 4–14, 2014.
194. M. Turk. "Multimodal interaction: A review". Pattern Recognition Letters, Volume 36, 15, Pages 189–195, 2014.

195. A. Revuelta-Martinez, L. Rodriguez, I. Garcia-Varea, F. Montero. “Multimodal Interaction for Information Retrieval using Natural Language”, Computer Standards and Interfaces, 35 (5), pp: 428–441, 2013.
196. L.H. Phuong, M. Visani, A. Boucher, J.-M. Ogier. “A new Interactive Semi-Supervised Clustering model for large image database indexing”. Pattern Recognition Letters, 2013.
197. C. Panagiotakis, H. Papadakis, E. Grinias, N. Komodakis, P. Fragopoulou, G. Tziritas. “Interactive image segmentation based on synthetic graph coordinates”. Pattern Recognition 46 (11), pp: 2940–2952, 2013.
198. P.A.V. de Miranda, X.A. Falcão, J.K. Udupa. “Synergistic arc-weight estimation for interactive image segmentation using graphs”. Computer Vision and Image Understanding 114(1): 85-99, 2010.
199. D. Holten, J.J.V. Wijk. “Visual comparison of hierarchically organized data”. Computer Graphics Forum 27, 2008.
200. K. Andrews, M. Wohlfahrt, G. Wurzinger. “Visual graph comparison”. IV'2009, IEEE, pp. 62–67, 2009.
201. C. Ahlberg. “Spotfire: An Information Exploration Environment”. SIGMOD Record 25(4): 25-29, 1996.
202. C. Erten, P.J. Harding, S.G. Kobourov, K. Wampler, G.V. Yee. “Graphael: Graph animations with evolving layouts”. Symposium on Graph Drawing. pp. 98–110, 2003.
203. S. Diehl, C. Gorg. “Graphs, they are changing dynamic graph drawing for a sequence of graphs”. Graph Drawing, 2002.
204. K. Yee, D. Fisher. “Animated exploration of dynamic graphs with radial layout”. INFOVIS. pp. 43–50, 2001.
205. C. Collins, M.S.T. Carpendale. “Vislink: Revealing relationships amongst visualizations”. IEEE Transactions on Visualization and Computer Graphics 13, 1192– 1199, 2006.
206. N. Elmqvist, P. Dragicevic, J.D. Fekete. “Rolling the Dice: Multidimensional Visual Exploration using Scatterplot Matrix Navigation”. IEEE TVCG, 14(6), 2008.
207. J.M. Kleinberg. “Bursty and hierarchical structure in streams”. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, ACM Press. p. 91-101, 2002.
208. M. Katevenis, S. Sidiropoulos, C. Courcoubetis. “Weighted round-robin cell multiplexing in a general-purpose ATM switch chip”. IEEE Journal on Selected Areas in Communications, (Vol. 9, Issue:8), 1991.
209. R. Hartley, A. Zisserman. “Multiple View Geometry in computer vision”. Cambridge University Press, 2003.
210. Forsyth, Ponce. “Computer Vision, a modern approach”. pp: 717, 1993.
211. E. W. Dijkstra. “A note on two problems in connexion with graphs”. Numerische Mathematik, 1, pp: 269–271, 1959.
212. N. Snavely, S. Todorovic. “From contours to 3D object detection and pose estimation”. International Congress on Computer Vision, 2011.

213. A. Rubio *et. al.*, "Efficient monocular pose estimation for complex 3D models". International Congress on Robotics and Automation, 2015.
214. S. Nene, S. Nayar, H. Murase. "Columbia Object Image Library: COIL-100". Technical report, Department of Computer Science, Columbia University, New York, 1996.
215. C. Harris, M. Stephens. "A combined corner and edge detection". Proceedings of The Fourth Alvey Vision Conference, pp. 147–151, 1988.
216. K. Borgwardt, C. Ong, S. Schönauer, S. Vishwanathan, A. Smola, H.P. Kriegel. "Protein function prediction via graph kernels". Bioinformatics 21(1), 47–56, 2005.
217. http://deim.urv.cat/~francesc.serratosa/databases/

Final Acknowledgment

This research has been supported by the TIN2013-47245-C2-2-R project and by the Universitat Rovira i Virgili through a PhD research grant.