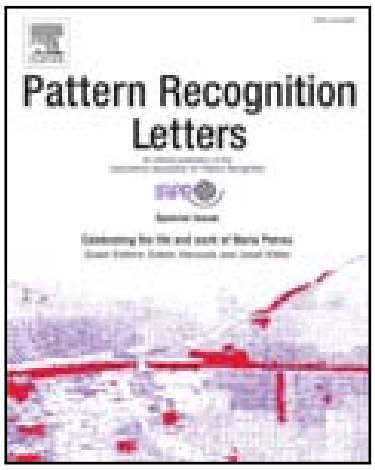
Accepted Manuscript



Obtaining the Consensus of Multiple Correspondences between Graphs through Online Learning

Carlos Francisco Moreno-García , Francesc Serratosa

PII: S0167-8655(16)30236-7
DOI: 10.1016/j.patrec.2016.09.003
Reference: PATREC 6637

To appear in: *Pattern Recognition Letters*

Received date: 27 October 2015
Accepted date: 2 September 2016

Please cite this article as: Carlos Francisco Moreno-García , Francesc Serratosa , Obtaining the Consensus of Multiple Correspondences between Graphs through Online Learning, *Pattern Recognition Letters* (2016), doi: 10.1016/j.patrec.2016.09.003

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Highlights

- Several graph extractors and graph matching algorithms return different correspondences.
- A method to deduce a consensus correspondence given several graph correspondences.
- A learning method to deduce the quality of each involved graph extractor and graph matching algorithm.
- The learnt quality is considered as the weight in the consensus algorithm.
- Sub-optimal solution. Cubic computational cost with respect to the number of nodes.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Obtaining the Consensus of Multiple Correspondences between Graphs through Online Learning

Carlos Francisco Moreno-García[a] and Francesc Serratosa[a]∗

[a]*Universitat Rovira i Virgili, Tarragona, Catalonia, Spain*

ABSTRACT

In structural pattern recognition, it is usual to compare a pair of objects through the generation of a correspondence between the elements of each of their local parts. To do so, one of the most natural ways to represent these objects is through attributed graphs. Several existing graph extraction methods could be implemented and thus, numerous graphs, which may not only differ in their nodes and edge structure but also in their attribute domains, could be created from the same object. Afterwards, a matching process is implemented to generate the correspondence between two attributed graphs, and depending on the selected graph matching method, a unique correspondence is generated from a given pair of attributed graphs. The combination of these factors leads to the possibility of a large quantity of correspondences between the two original objects. This paper presents a method that tackles this problem by considering multiple correspondences to conform a single one called a consensus correspondence, eliminating both the incongruences introduced by the graph extraction and the graph matching processes. Additionally, through the application of an online learning algorithm, it is possible to deduce some weights that influence on the generation of the consensus correspondence. This means that the algorithm automatically learns the quality of both the attribute domain and the correspondence for every initial correspondence proposal to be considered in the consensus, and defines a set of weights based on this quality. It is shown that the method automatically tends to assign larger values to high quality initial proposals, and therefore is capable to deduce better consensus correspondences.

Keywords: Attributed graphs, graph matching, consensus correspondence, online learning

2015 Elsevier Ltd. All rights reserved.

ACCEPTED MANUSCRIPT

∗ Corresponding author. e-mail: francesc.serratosa@urv.cat

1. Introduction

Structural pattern recognition has the aim of representing, identifying and classifying objects such as images, handwritten characters, biometric data, networks or proteins [1], among many others. These objects are transformed into structures which are constituted of properly interconnected parts. There are numerous forms in which these data representations can be stored, such as strings, trees or data clusters. Remarkably, the most recurrent form of data representation used nowadays is attributed graphs. Given two or more attributed graphs, the identification or the classification process is typically performed by establishing a one-to-one relation function between the parts of the two graphs. This process, most commonly known as *graph matching*, has been widely studied during the last 4 decades and compiled in surveys such as [2], [3] and [4].

To build the attributed graphs in the first place, a graph extraction process is applied to the pair of objects to be matched. That is, selecting the most representative features of the objects as the nodes of the graph, and assigning them a set of labels called attributes. In the concrete case of images, this is done by using a Feature Extractor (FE) [5] to select the nodes, and then applying an edge construction method to such set of nodes. Afterwards, a correspondence between two graphs can be deduced by mapping the nodes and/or edges through a graph matching algorithm. Due to the wide availability of FEs (i.e. SURF [6] or SIFT [7]), edge construction methods (i.e. Delaunay or k-nearest neighbours) and graph matching algorithms (i.e. Graduated Assignment [8], Bipartite (BP) graph matching [9], Fast BP (FBP) [10], or Squared FBP (SFBP) [11]), multiple combinations of graphs and correspondences between them could be generated from a single pair of objects. Therefore, the aim of this paper is to present a method in which a consensus correspondence can be generated give multiple graph matching solutions.

The most basic form to introduce this problem is through an image matching scenario for the case that two different FEs and edge construction methods are used to extract the salient points from the images, and then two graph matching algorithms are applied. Using the first FE, both objects are represented through attributed graphs G^1 and G'^1 (red circles in the left hand side of Figure 1.a). A similar operation with the second FE leads to the creation of graphs G^2 and G'^2 (blue squares in the right hand side of Figure 1.a). It is clearly noticeable that there are some features that have been selected equally by both FEs, and thus the intersection between red circles and blue squares is not null. Figure 1.b shows the two correspondences f^1 and f^2 that have been generated between both pairs of graphs. Notice that the two proposals present discrepancies not only on the features of each graph, but also in the node-to-node mappings. Moreover, a new node called null node (represented with the Φ symbol) has been introduced to assure that correspondence f^1 is bijective. Finally, Figure 1.c shows a possible consensus correspondence $f^{1,2}$, where $G^{1,2}$ and $G'^{1,2}$ represent the union of nodes of G^1 and G^2, and the union of nodes of G'^1 and G'^2 respectively. Edges are not set in these final sets since the discrepancies between the edges of the initial graphs would lead to edge inconsistencies. Furthermore, notice that the aim of our proposed consensus solution is not to produce a union graph (methods such as function-described graph [12], second-order random graph [13] or median graph [14], [15] achieve this), but rather to define a consensus correspondence between the union of nodes of both sets of graphs.

In general, the input of the consensus method consists on N bijective correspondences, $f^1: G^1 \rightarrow G'^1, \ldots, f^k: G^k \rightarrow G'^k, \ldots, f^N: G^N \rightarrow G'^N$, with the output being the consensus correspondence $f: G^{1,\ldots,k,\ldots,N} \rightarrow G'^{1,\ldots,k,\ldots,N}$. It is important to comment that the method requires that nodes within either sets of graphs $G^1, \ldots, G^k, \ldots, G^N$ or $G'^1, \ldots, G'^k, \ldots, G'^N$ are indexed in such manner that nodes on different graphs, but with the same index, represent the same local part of the object. For instance, in Figure 1.a, the node of G^1 (red circle) located on the subject's knee must have the same index as the node on G^2 (blue square) located on the same section.

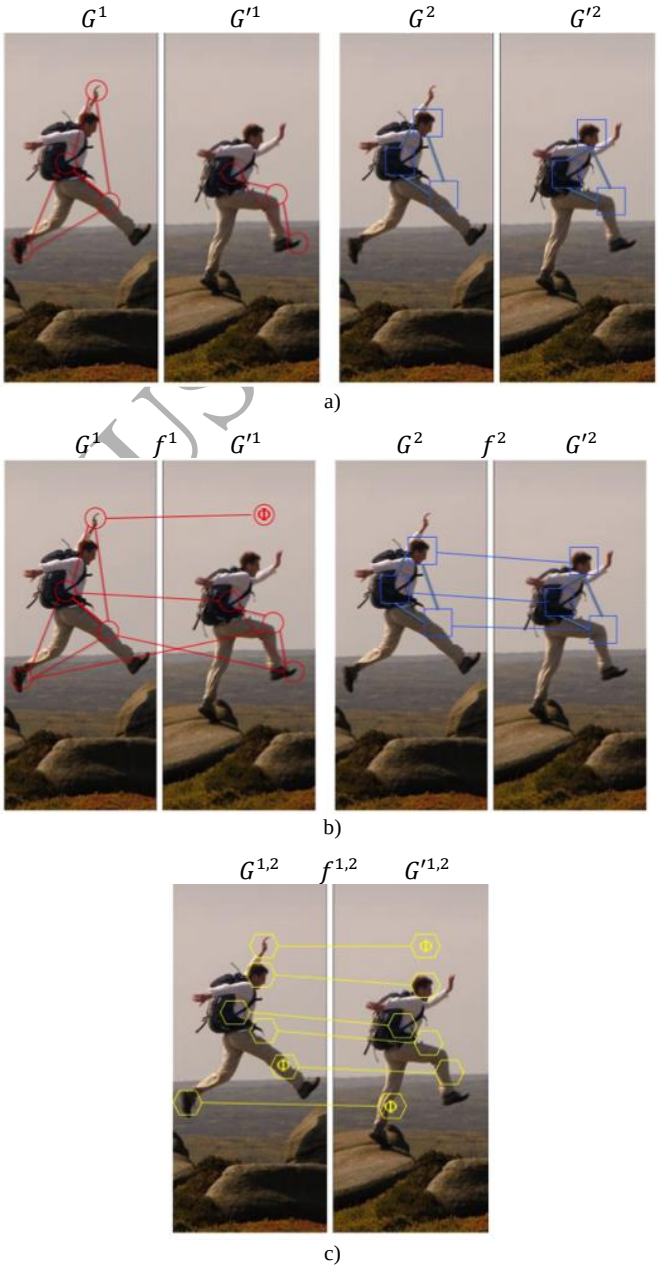


Fig. 1. An example of obtaining a consensus correspondence given two initial proposals.

An immediate problem that arises when intending to solve this problem is that a large number of possible consensus correspondences can be deduced. To solve this issue, one of the most well-known and practical options to reduce the complexity of a combinatorial calculation is through optimisation strategies. The concept of optimisation is related to the selection of the "best" configuration or set of parameters to achieve a certain goal. Functions involved in an optimisation problem can be either conformed by continuous or discrete values, often called

ACCEPTED MANUSCRIPT

combinatorial scenarios. These scenarios have been largely studied and applied for matching problems, particularly in the case of the Hungarian algorithm [16] and the Jonker-Volgenant solver [17]. Both of these methods convert a combinatorial problem into a Linear Assignment Problem (LAP). In fact, the BP graph matching [9], one of the aforementioned algorithms to obtain a correspondence between attributed graphs based on the Graph Edit Distance (GED) [18], [19], [20], [21], could also be used for the solution of the LAP. This algorithm computes a cost matrix, and then obtains the correspondence by applying a linear solver such as [16] or [17]. Therefore, the consensus method proposed in this paper can be considered a generalisation of the BP graph matching algorithm [9], but instead of having a pair of graphs as the inputs, our method deals with multiple pairs of attributed graphs and their node-to-node correspondences. Recently, some collaborative methods have been proposed in which, given a set of classifiers, they return the most promising class [22]. These methods learn some weights that gauge the importance of both the classifiers and the samples through several techniques, such as voting [23] or hierarchical methods [24]. Nevertheless, these methods cannot be directly adapted to our problem, since their output is a class index, whereas our required output must be a consensus correspondence.

In recent years, we have developed a series of approaches to solve similar correspondence-based consensus scenarios. In [25] and [26], we presented a method that computes the consensus correspondence given only two pairs of sets of salient points and two correspondences between them. Later, in [27] and [28], we generalised this method for multiple pairs of sets of salient points and the correspondences between them. In [29], we redefined the method presented in [25] and [26] so that instead of sets of salient points, it considers a pair of attributed graphs and two correspondences between them. As a logical next step, in this paper, we propose to expand the reach of the method presented in [29], and thus, the input is composed of multiple pairs of graphs and the correspondences between them. In addition, we improve the consensus methodology through an algorithm presented in [30] that learns a set of weights that gauge the quality of both the attributed graphs and the correspondences between them. Furthermore, in the experimental section we aim to show that properly tuning these weights is crucial to obtain an improved consensus.

The paper is structured as follows. In Section 2, we briefly define the concepts of attributed graphs, GED and BP graph matching. In Section 3, we explain the methodology to obtain the consensus correspondence given several pairs of attributed graphs and their correspondences. Section 4 complements this explanation by revealing how the weights are learnt. Section 5 shows the experimental evaluation and finally, Section 6 concludes the paper.

2. Attributed Graphs, Graph Edit Distance and Bipartite Graph Matching

Let $G^1 = (\Sigma_v^1, \Sigma_e^1, \gamma_v^1, \gamma_e^1)$,…,$G^k = (\Sigma_v^k, \Sigma_e^k, \gamma_v^k, \gamma_e^k)$,…, $G^N = (\Sigma_v^N, \Sigma_e^N, \gamma_v^N, \gamma_e^N)$ be attributed graphs representing a first object, and $G'^1 = (\Sigma'^1_v, \Sigma'^1_e, \gamma'^1_v, \gamma'^1_e)$,…,$G'^k = (\Sigma'^k_v, \Sigma'^k_e, \gamma'^k_v, \gamma'^k_e)$,…, $G'^N = (\Sigma'^N_v, \Sigma'^N_e, \gamma'^N_v, \gamma'^N_e)$ be attributed graphs representing a second object. To allow maximum flexibility in the matching process, all of these graphs have been extended with null nodes for all of them to be of order M. Then, $\Sigma_v^k = \{v_i^k \mid i = 1, \dots, M\}$ represents the set of nodes and $\Sigma_e^k = \{e_{x,y}^k \mid x, y \in 1, \dots, M\}$ represents the set of edges. Also, functions $\gamma_v^k: \Sigma_v^k \to \Delta_v^k$ and $\gamma_e^k: \Sigma_e^k \to \Delta_e^k$ assign attribute values in any domain to nodes and edges respectively. Note that attributes on nodes and edges can have different values, that is, it may happen that $\Delta_v^k \neq \Delta_v^t$ and $\Delta_e^k \neq \Delta_e^t$. Nevertheless, all of these domains have the same value Φ defined, which represents the attribute of the null nodes used for the extension. Similar properties are preserved for the graphs representing the second object. As commented before, nodes are indexed in such manner that the nodes on different graphs, but with the same index, represent the same local part of the object. If a node does not exist in the original object, then it is inserted with the null attribute Φ.

GED [18], [19], [20], [21] is the most well-known and used distance function between attributed graphs. It is defined as the minimum amount of required distortion to transform one graph into another. To this end, a number of distortion or edit operations consisting of deletion, insertion, and substitution of nodes and edges are defined. These edit cost functions are introduced to quantitatively evaluate the edit operations. The basic idea is to assign a penalty cost to each edit operation according to the amount of distortion that it introduces in the transformation. Deletion (insertion) operations are transformed into assignations of a non-null node (null node) to a null node (non-null node). Substitutions simply indicate node-to-node assignations.

Given two attributed graphs G^k and G'^k, and a correspondence f^k between them, the graph edit cost, represented by the expression $EditCost(G^k, G^{k\prime}, f^k)$, is the cost of the edit operations that the correspondence imposes. It is based on adding the following constants and functions:

- C_{vs} is a function that represents the cost of substituting node v_i^k of G^k by node $f^k(v_j^k)$ of G'^k.
- C_{es} is a function that represents the cost of substituting edge $e_{i,j}^k$ of G^k by edge $f^k(e_{i,j}^k)$ of G'^k.
- Constant K_v is the cost of deleting node v_i^k of G^k(mapping it to a null node) or inserting node v'^k_j of G'^k (being mapped from a null node).
- Constant K_e is the cost of assigning edge $e_{i,k}^k$ of G^k to a null edge of G'^k, or assigning edge $e'^k_{i,j}$ of G'^k to a null edge of G^k.

For the cases in which two null nodes or two null edges are mapped, this cost is 0. Then, the GED is defined as the minimum cost under any possible bijective correspondence T:

$$GED(G^k, G'^k) = \min_{f^k \in T}\{EditCost(G^k, G'^k, f^k)\}. \quad (1)$$

We define the optimal correspondence f^* as the one that obtains the minimum $EditCost$.

One of the most used algorithms to compute error-tolerant graph matching through the application of the GED is the BP graph matching algorithm [9]. It is composed of three main steps. First, a cost matrix is defined. Second, a linear solver such as the Hungarian method [16] or the Jonker-Volgenant solver [17] is applied to this matrix to obtain the optimal correspondence f^*. Finally, a suboptimal GED value is obtained from the edit cost of the resulting correspondence, that is $EditCost(G^k, G'^k, f^*)$. Figure 2 shows the cost matrix defined for the BP algorithm.

The first quadrant (top-left) denotes the combination of substituting costs $C_{i,j}$ and their local sub-structures. The diagonal of the second quadrant (top-right) denotes the whole costs $C_{i,\varepsilon}$ of

ACCEPTED MANUSCRIPT

deleting nodes v_i^k and its local sub-structures. Likewise, the diagonal of the third quadrant (bottom-left) denotes the whole costs $C_{\varepsilon,j}$ of inserting nodes v'^k_j and their local sub-structures. Finally, the fourth quadrant (bottom-right) is filled with zero values, since the substitution between null nodes has a zero cost. To establish the costs used on the BP matrix, it is usual to consider the attributes of the local sub-structure of each node. Currently, the most used local sub-structures are Node, Degree and Star (also called Clique) [31], [32].

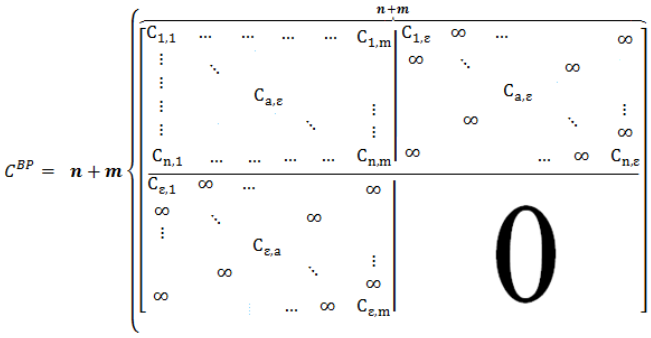


Fig. 2. Cost matrix of the BP algorithm.

3. Consensus of Multiple Correspondences between Graphs

Given a set of output graphs $G^1, \dots, G^k, \dots, G^N$ and a set of input graphs $G'^1, \dots, G'^k, \dots, G'^N$, together with a set of bijective correspondences $f^1, \dots, f^k, \dots, f^N$ between them, it is intended to deduce a final consensus correspondence f. It is assumed that $\Delta_v^k = \Delta'^k_v$ and $\Delta_e^k = \Delta'^k_e$; however, it may happen that $\Delta_v^k \neq \Delta_v^q$, $\Delta'^k_v \neq \Delta'^q_v$, $\Delta'^k_e \neq \Delta'^q_e$ or $\Delta_e^k \neq \Delta_e^q$. Moreover, it is assumed that there is some level of intersection between the nodes of graphs within the same set, although this is not strictly necessary. This intersection is defined through the indices of the nodes in both sets of graphs, since as commented previously, two nodes with the same index value represent the same local part of an object. In addition, all graphs are extended with null nodes to have the same order M. The output of the consensus method is a correspondence $f: \Sigma_v \rightarrow \Sigma'_v$, where Σ_v is composed of the nodes that belong to sets $\Sigma_v^1, \dots, \Sigma'^k_v, \dots, \Sigma_v^N$ that have different indices. Coherent definitions hold for Σ'_v.

It is proposed that the consensus correspondence f holds two restrictions. On the one hand, the edit costs obtained by f for all pairs of graphs has to be the lowest possible, that is

$$\min_{f \in T} \sum_{k=1}^{N} EditCost(G^k, G'^k, f). \tag{2}$$

This term is used since by definition of the GED, the lower the edit cost, the better the correspondence. On the other hand, it is necessary to take into consideration the contribution of $f^1, \dots, f^k, \dots, f^N$ to a certain extent. Therefore, the consensus correspondence must somehow resemble the original correspondences as much as possible. For this to occur, the Hamming distances between the consensus correspondence and the original correspondences have to be minimised,

$$\min_{f \in T} \sum_{k=1}^{N} d_H(f^k, f). \tag{3}$$

In the final expression, both Equation 2 and Equation 3 are weighted and added, thus resulting in

$$\min_{f \in T} \sum_{k=1}^{N} \alpha^k \cdot d_H(f^k, f) + \beta^k \cdot EditCost(G^k, G'^k, f). \tag{4}$$

Equation 4 can be seen as a classical combinatorial optimisation solution, composed of a loss function and a regularisation term [33]. In this case, the loss function is represented by Equation 2, since this expression aims at minimising the cost of the consensus correspondence to the most. Conversely, the regularisation term is represented by the Equation 3, which has the aim of finding a correspondence resembling the original ones. Notice that it would not be adequate to use methods based on the gradient descent, such as the Nelder-Mean [34], to minimise Equation 4, given that a simple swap of any node-to-node mappings on the consensus correspondence could cause an abrupt change on both the Hamming distance and the edit cost, since both terms are non-continuous. Moreover, if methods such as simulated annealing or genetics are considered, then the condition of the consensus correspondence to be bijective is not guaranteed. Therefore, we propose a generalisation of the BP algorithm to solve this equation. The main difference between the original BP algorithm and the consensus method proposed in this paper resides in the definition of the cost matrix on the first step. For the consensus method, both the cost of the attributed graphs and the cost of the correspondences are used as the inputs to generate the final matrix.

The general structure of the matrix used on the consensus method is also composed of four quadrants, as shown in Figure 3. The top-left quadrant is reserved for the mappings between nodes in graphs G^k and G'^k. The top-right quadrant represents the mapping deletions. Conversely, the bottom-left quadrant represents mapping insertions. Finally, the bottom-right quadrant is reserved for mappings between null nodes and thus, is filled with zero values, in the same way as the original BP cost matrix.

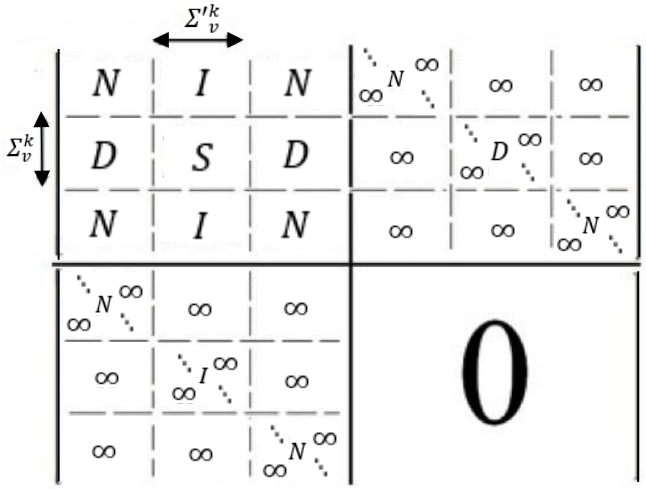


Fig. 3. Distribution of rows and columns in the matrix used on the consensus method.

It is possible to distinguish four different types of cells in this matrix: 1) Type S. Represents the case where both nodes are non-null nodes. 2) Type D. Represents the case where the node in the output graph is a non-null node, and the node in the input graph is a null node. 3) Type I. Represents the opposite of the previous case. 4) Type N. Represents the case where both nodes are null.

In the BP algorithm cost matrix (Figure 2), every cell on the top-left quadrant would be of type S. In contrast the new cost matrix (Figure 3) presents type D and type I cells in this same quadrant, which is derived from the fact that nodes have been added to the graphs for all correspondences to be mutually bijective and for all graphs to be of order M. This aspect is illustrated on the example presented in Figure 1.b, where the mapping towards the node Φ of the red graph represents information contained in a type D cell of the top-left quadrant. Independently, both the type D cells in the top-right quadrant and the type I cells in the bottom-left quadrant are a derivation of the BP algorithm, and allow new nodes to be inserted or deleted.

Since the consensus method considers two constraints given several $1, \dots, k, \dots, N$ pairs of graphs and correspondences between them, the final matrix to be minimised is composed by the addition of F^k correspondence matrices plus C^k cost matrices, gauged by weights α^k and β^k respectively. As a result, the final matrix H is built as follows,

$$H = \sum_{k=1}^{N} -F^k \cdot \alpha^k + C^k \cdot \beta^k. \qquad (5)$$

Figure 4 shows an example of the structure of a F^k correspondence matrix. We set $F^k[i,j] = 1$ if $f^k(v_i) = v'_j$ and $F^k[i,j] = 0$ otherwise. This results in only the S type cell of the first quadrant being filled with either zeros or ones according to the information provided by the correspondence. The second and third quadrants are filled with zero values, since they represent the possibility of deleting or inserting new nodes; which is not the case for the information provided by a correspondence f^k. Notice that correspondence matrix F^k is put together with a negative sign when calculating matrix H in Equation 5. This is done since the minimisation of H requires that existing mappings $f^k(v_i) = v'_j$ are represented in the correspondence matrix F^k with lower values.

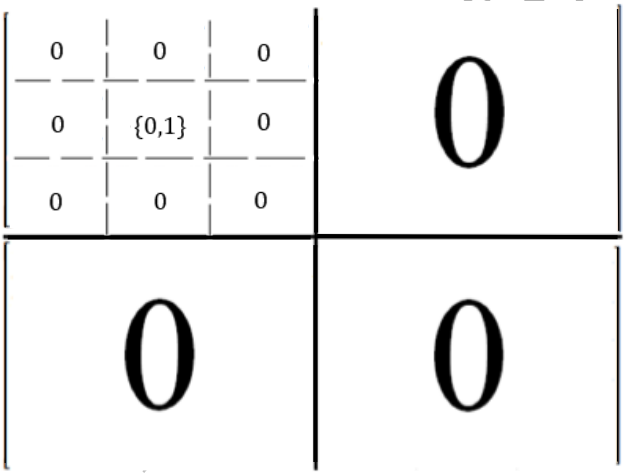


Fig. 4. Correspondence matrix $\boldsymbol{F^k}$.

Figure 5 shows the structure of a cost matrix C^k. The type S cells from Figure 3 are filled with the node substitution cost $C^k_{i,j}$. The type D and type I cells are filled with insertion and deletion costs $C^k_{i,\varepsilon}$ and $C^k_{\varepsilon,j}$ respectively. Finally, type N cells which represent mappings between null nodes are not filled, since mappings of this nature would have a cost of 0. Since costs with lower values are preferred for the minimisation of H, the cost matrix C^k is used with a positive sign in Equation 5.

The final consensus correspondence between nodes is the one that achieves the minimum LAP solution applied to matrix H (Equation 5), using any solver such as the Hungarian method [16] or the Jonker-Volgenant solver [17]. Algorithm 1 shows the steps followed to compute the consensus correspondence.

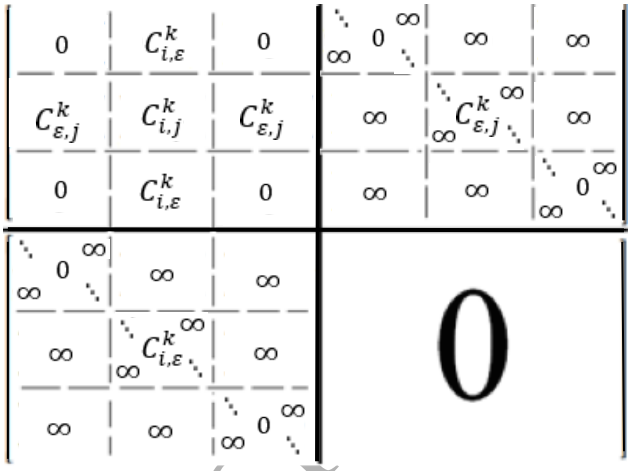


Fig. 5. Cost matrix $\boldsymbol{C^k}$.

The computational cost of the whole parameter calculation is $O(N \cdot M^2)$, and the computational cost of the linear solver is $O(M^3)$. In the next section, we explain how function $Online\_Learning$ learns parameters α^k and β^k.

Algorithm 1. $Consensus\_Method$
Input:
$\{f^1, \dots, f^k, \dots, f^N\}, \{G^1, \dots, G^k, \dots, G^N\}, \{G'^1, \dots, G'^k, \dots, G'^N\}$
Output: f
for $\boldsymbol{k = 1:N}$
  for all $\boldsymbol{i, j}$
    $F^k[i,j] = 1 \;\; if \;\; f^k(v_i^k) = v'^k_j$
    $F^k[i,j] = 0 \;\; otherwise.$
  end for
$C^k = Compute\_Cost\_Matrix(G^k, G'^k)$
$(\alpha^k, \beta^k) = Online\_Learning\{Learning\_Set[Q^k]\}$
end for
$H = \sum_{k=1}^{N} -F^k \cdot \alpha^k + C^k \cdot \beta^k$
$f = LinearSolver(H)$
End algorithm

4. Online Learning the Quality

In this section, we describe how to learn the weighting parameters $\alpha = (\alpha^1, \dots, \alpha^k, \dots, \alpha^N)$ and $\beta = (\beta^1, \dots, \beta^k, \dots, \beta^N)$. These weights represent respectively the level of confidence that the consensus method should have on the combination of the graph matching methods (represented by correspondence matrices $F^1, \dots, F^k, \dots, F^N$) and the graph extraction methods (represented by cost matrices $C^1, \dots, C^k, \dots, C^N$) used to compute each of the $1, \dots k, \dots, N$ correspondences. It is assumed that if a certain F^k correspondence matrix or its respective C^k cost matrix are properly defined, then the values of their respective parameters α^k or β^k have to be high. That is, either f^k represents a correct correspondence, or the distance between features $C^k_{i,j}$ really reflects the dissimilarity between v_i^k and v'^k_j and their local sub-structures. Notice that the learning algorithm is defined in an online form, meaning that it does not force the user to impose the values simultaneously and thus, it is suitable for applications [35] where the data is not available at the same time. As a result, each of the k^{th} elements of α^k and β^k are updated separately.

To learn the weights, it is necessary to define a learning set composed of multiple registers $1, \dots, t, \dots, T^k$, where T^k is the number of available registers for each k proposal. Each of these registers is a quartet with structure $\{G_t^k, G'^k_t, f_t^k, \check{f}_t^k\}$, where G_t^k and G'^k_t are attributed graphs, f_t^k is a sub-optimal correspondence between them, and $\check{f}_t^k$ is an optimal ground truth correspondence between these graphs.

Both sets of weights are computed as follows. First, weights α^k gauge the quality of the correspondence. These are calculated as a similarity function between the correspondence f_t^k and the ground truth correspondence $\check{f}_t^k$. The similarity is calculated as the inverse of the Hamming distance. Note the obtained value does not depend on the graphs G_t^k and G'^k_t, but only on the correspondences. Secondly, weights β^k gauge the quality of the features. This step considers the $C_{i,j}^k$ cost between elements, and also the ground truth $\check{f}_t^k$. Note that in this case, the sub-optimal correspondence f_t^k is not used. We consider as the genuine cells in $C_{i,j}^k$ the ones such that $\check{f}_t^k(a_i^k) = a'^k_j$, and as the impostor cells the remaining ones, $\check{f}_t^k(a_i^k) \neq a'^k_j$. With this information, two histograms $H_{genuine}^k$ and $H_{impostor}^k$ are constructed. As more distant the two histograms are, the better these features are represented on the ground truth correspondence. Therefore, β^k is defined as the distance between these two histograms, and this distance is computed through the Earth movers' distance [36] between histograms. We decided to use this distance instead of the typical Mahalanobis distance between probability density functions, because in the first samples, the approximation error was very high.

Function 1 computes weights α^k and β^k for a certain number of $0, \dots, t, \dots, Q^k$ registers of the k^{th} proposal contained on the learning set, represented as $Learning\_Set[Q^k]$. For instance, if every register of the learning set is used, then $Q^k = T^k$. Notice each time this function is computed, parameters Q_{acc}^k S^k, $H_{genuine}^k$ and $H_{impostor}^k$ are updated. Q_{acc}^k is the cumulative number of registers that have been used to compute the weights so far, while S^k is the cumulative value of the similarity (while computing this value, a 1 is added in the denominator to avoid the infinite case), and $H_{genuine}^k$ and $H_{impostor}^k$ are the cumulative histograms. Before the function is called for the first time, parameters for all k are initialised as follows: $\alpha^k = 1/N$, $\beta^k = 1/N$, $Q_{acc}^k = 0$, $S^k = 0$, $H_{genuine}^k = 0$ and $H_{impostor}^k = 0$.

Function 1. $Online\_Learning$
Input: $Learning\_Set[Q^k], \alpha^k, \beta^k, Q^k, H_{genuine}^k, H_{impostor}^k, S^k$
Output: $\alpha^k, \beta^k, Q_{acc}^k, H_{genuine}^k, H_{impostor}^k, S^k$

$Q_{acc}^k = Q_{acc}^k + Q^k$
for $\boldsymbol{t = 1}$: $\boldsymbol{Q^k}$
$C_t^k = Compute\_Cost\_Matrix(G_t^k, G'^k_t)$
$H_{genuine}^k = H_{genuine}^k + histogram_{genuine}(C_t^k)$
$H_{impostor}^k = H_{impostor}^k + histogram_{impostor}(C_t^k)$
$S^k = S^k + \frac{1}{HammingDistance\left(f_t^k, \check{f}_t^k\right)+1}$
end for

$$\alpha^k = \frac{S^k}{Q_{acc}^k}$$

$$\beta^k = \frac{EarthMoversDistance\left(H_{genuine}^k, H_{impostor}^k\right)}{Q_{acc}^k}$$

End function

5. Experimental Validation

The following section is organised in three subsections. First, in Section 5.1 we describe the database used for experimentation. Then, in Section 5.2 we present the results of the consensus framework for the case when no learning is applied. Finally, Section 5.3 show the results and comparison of applying the consensus framework considering the learning weights function.

5.1. Database Used

We used the "Tarragona Exteriors" database [37], defined through five public image sequences called "BOAT", "EAST_PARK", "EAST_SOUTH", "RESIDENCE" and "ENSIMAG" [38]. These datasets are composed of sequences of 11 images taken from the same object, but from different positions and using a different zoom. Together with the images, the homography estimations that convert the first image of the set into the other ones are provided. From each of the images, the 50 most reliable salient points were extracted using 5 FEs: FAST [39], HARRIS [40], MINEIGEN [41], SURF [6] and SIFT [7]. From these five sets of salient points, we built five attributed graphs of each image, where the nodes represent the position of the salient points, and the edges are conformed using the Delaunay triangulation method. Attributes on nodes are the attribute vectors provided by each FE, and edges are unattributed.

Between the first image of the sequence and the other ten images (10 image pairings), we generated correspondences using the five different attributed graphs in combination with four different matching approaches: 1) MATCH: the native Matlab matching function [42] called *MatchFeatures*. 2) BP-NODE: SFBP with the Node local structure (no edges). 3) BP-DEGREE: SFBP with the Degree local structure (a central node and its adjacent edges). 4) BP-CLIQUE: FBP [10] with the Clique structure (a central node, the adjacent edges and their connected nodes). The *MaxRatio* parameter of the *MatchFeatures* function was set to 1 to find as many mappings as possible, and the non-bijective correspondences were removed afterwards, since this function often maps output nodes more than once. Moreover, parameters K_v and K_e of the SFBP algorithm were set to 50 for both sub-structures, and to 250 in FBP algorithm for the CLIQUE sub-structure. The substitution cost on nodes C_{vs} is the normalised Euclidean distance between the attribute vectors, and the substitution cost on edges is $C_{es} = 0$, due to edges being unattributed. The ground truth correspondences were computed with the homographies provided on the original image databases.

As a result, the "Tarragona Exteriors" database is composed of 20 datasets. Each dataset is the combination of the 5 image sequences × 4 graph matching methods. Each dataset is composed of 50 registers with structure $\{G^k, G^{k\prime}, f^k, \check{f}^k\}$. Due to the 5 FEs, $1 \leq k \leq 5$. FAST is represented by $k = 1$, HARRIS: $k = 2$, MINEIGEN: $k = 3$, SURF: $k = 4$ and SIFT: $k = 5$. Furthermore, there are 10 different registers $\{G^k, G^{k\prime}, f^k, \check{f}^k\}$ per each k.

Table 1 shows the average number of correct mappings obtained by the correspondences contained in the 20 initial datasets, with the last column showing the average values. In 14 of the cases, the combinations that obtains the highest number of correct mappings are the ones where the SURF FE is used. In the case of MATCH, this observation is always true, since the *matchFeatures* function is specifically designed to be used with this FE. Observing the average results, correspondences generated through the BP-CLIQUE graph matching algorithm obtain better results. This is because by using the clique local

sub-structure, more structural information is captured and thus, the graph matching algorithm performs better.

5.2. Consensus of two correspondences without learning

Table 2 shows the average number of correct mappings obtained with the consensus method presented in [29], where only two proposals are used at the same time, that is $N = 2$. Each column is labelled as $E_{k,q}$, where k and q are the two registers used, differentiated by their respective FE. Weights have not been learned and are set to 0.5. Considering the information provided by Table 1, we realise that combinations using SURF lead to the best results. For instance, for all datasets where the *matchFeatures* function has been used, combinations where the SURF FE appears ($E_{1,4}$, $E_{2,4}$, $E_{3,4}$ and $E_{4,5}$) obtain also the best results. Moreover, in 4 out of the 5 sequences, the BP-CLIQUE shows the highest improvement. Comparing these two tables, we conclude that in all experiments, the combination of two FEs increases the individual number of inlier mappings found. Therefore, it is shown that the consensus method generates consensus correspondences which increase in quality even when no learning has been applied.

5.3. Multiple consensus plus learning weights

Before validating the results of the consensus method, we present the progression of learning parameters α^k and β^k. To do so, we only consider the first four quartets of each dataset and k as the learning set: $\{G_1^k, G'^k_1, f_1^k, \acute{f}_1^k\}, \dots, \{G_4^k, G'^k_4, f_4^k, \acute{f}_4^k\}$. Therefore, the online algorithm is able to iterate 4 times per each k and dataset. Figure 6 and Figure 7 show the evolution of α^k and β^k for $Q^k = 0, \dots, t, \dots, 4$ learning iterations. Iteration $Q^k = 0$ means that no learning has taken place, therefore the weight values have been imposed as $\alpha^k = 0.2$ and $\beta^k = 0.2$. The values in these plots correspond to the average of the 20 datasets according to each FE. We have considered only 4 iterations since we have observed that after this point, plots tend to stabilise.

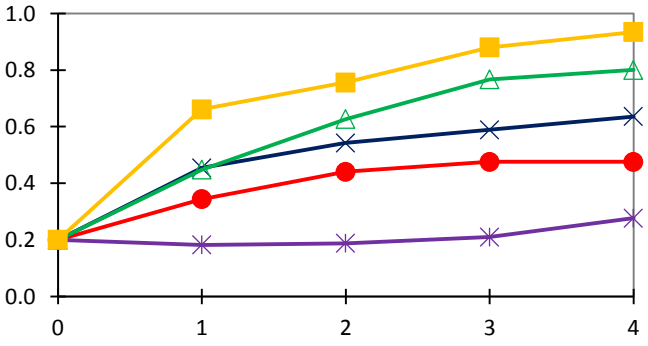


Fig. 6. Evolution of α^k for different Q^k learning iterations. (●)α^1= FAST, (×)α^2=HARRIS, (Δ)α^3=MINEIGEN, (■)α^4=SURF, (∗)α^5=SIFT.

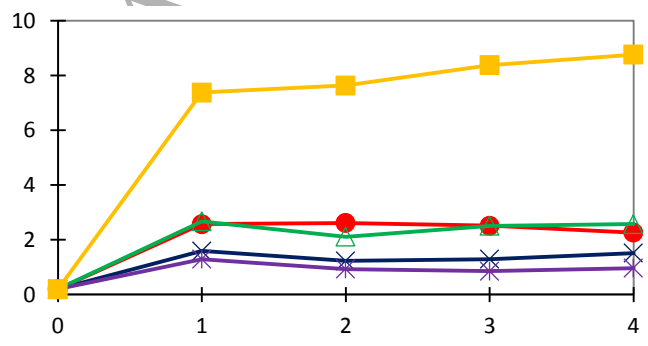


Fig. 7. Evolution of β^k for different Q^k learning iterations. (●)α^1= FAST, (×)α^2=HARRIS, (Δ)α^3=MINEIGEN, (■)α^4=SURF, (∗)α^5=SIFT.

There is a clear correlation between weights reported in Figure 6 and Figure 7 and the number of correct inliers reported for the original proposals in Table 1. For all datasets, it is shown that SURF is the FE that obtains the best results. Consequently, the learning algorithm automatically tends to assign higher values for weights α^4 and β^4, which are the ones related to SURF. If more learning iterations are considered, then the tendency of these lines is dependent of the information at hand. Therefore, the convergence of this algorithm cannot be demonstrated.

Table 3 shows the average number of correct mappings found on the consensus correspondence of all proposals for a same pair of images. The first two columns show the average number of correct mappings extracted directly from Table 1 and Table 2 respectively. The last column shows the number of correct mappings detected by the consensus method with the implementation of the online algorithm using $Q^k = 4$ learning iterations. Notice that for all cases, the final consensus method is able to increase the average accuracy with respect to the original results. Moreover, it is still noticeable that the BP-CLIQUE registers deliver the best results.

6. Conclusions

Most structural pattern recognition methods are based on first creating a representation of the objects, usually through attributed graphs, and then finding a correspondence between them using graph matching algorithms. Clearly, there are several solutions in the literature to either perform the graph extraction process or the graph matching process. Instead of only relying on one combination, in this paper we propose a method that globally considers multiple correspondences regardless of the path followed to obtain them. Since it is assumed the noise randomly affects the data in a non-repetitive way, the experimental validation shows the consensus method tends to obtain a better correspondence than the individual ones. Moreover, we implement a learning algorithm that automatically learns the weights that represent the quality of both the attributes of the graphs and the correspondences themselves. These weights are used to gauge the inputs that deliver the consensus correspondence. The experimental validation shows that properly learning these weights is crucial to achieve a large number of correct inlier mappings.

The BP graph matching algorithm has a cubic-worst computational complexity with respect to the number of nodes due to the LAP solver. Since our consensus method is a generalisation of the BP graph matching algorithm, it also has the same complexity. The only difference between our method and the classical one is the generation of the final matrix to be evaluated by the LAP solver. In the BP graph matching algorithm, this matrix is computed in square cost with respect to the number of nodes. In our method, we need to compute as many matrices as the number of different proposed correspondences. Therefore, the computational complexity of this step is the complexity in BP multiplied by the number of proposed correspondences.

As future work, we are interested in testing our consensus method using more databases, graph extraction methods and matching algorithms. In addition, it is desired to test with correspondences generated from other pairs of data structures, such as strings or data clusters.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table 1. Average number of correct mappings on the 20 datasets (in bold the highest values).

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

Table 2. Average number of correct mappings obtained by the consensus method using 2 input correspondences and without learning weights, for the 20 datasets.

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

Table 3. Average number of correct mappings for the individual extractors, the consensus method with 2 correspondences (without learning) and the consensus method using all correspondences and learning weights.

Dataset	Name	C1	C2	C3
1	BOAT_BP-CLIQUE	49.8	90.2	234
2	BOAT_BP-DEGREE	40.8	73.1	222
3	BOAT_BP-NODE	39.2	63.8	192
4	BOAT_MATCH	18	26.6	99
5	EASTPARK_BP-CLIQUE	26.6	55	137
6	EASTPARK_BP-DEGREE	23.4	33.8	117
7	EASTPARK_BP-NODE	19.6	32.9	112
8	EASTPARK_ MATCH	14.4	20.8	63
9	EASTSOUTH_BP-CLIQUE	8.4	18	32
10	EASTSOUTH_BP-DEGREE	10	18.3	31
11	EASTSOUTH_BP-NODE	7.8	11.9	21

ACCEPTED MANUSCRIPT

12	EASTSOUTH_ MATCH	8	10.2	19
13	RESIDENCE_BP-CLIQUE	31.4	63.2	147
14	RESIDENCE_BP-DEGREE	14.8	22.6	122
15	RESIDENCE_BP-NODE	11.8	17.6	107
16	RESIDENCE_ MATCH	22	26.2	78
17	ENSIMAG_BP-CLIQUE	10.8	21.3	59
18	ENSIMAG_BP-DEGREE	9	16.2	43
19	ENSIMAG_BP-NODE	8.2	15.8	27
20	ENSIMAG_ MATCH	11.4	17.5	24

Acknowledgments

This research is supported by the Spanish project DPI2013-42458-P, by project TIN2013-47245-C2-2-R and by Consejo Nacional de Ciencia y Tecnologías (CONACyT Mexico).

References

1. Riesen, K., Bunke, H., "IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning," SSPR, pp. 287–297, 2008.
2. Conte, D., Foggia, P., Sansone, C., Vento, M., "Thirty Years of Graph Matching in Pattern Recognition," International Journal of Pattern Recognition and Artificial Intelligence 18 (3), pp. 265–298, 2004.
3. Foggia, P., Percannella, G., Vento, M., "Graph Matching and Learning in Pattern Recognition in the Last Ten Years". International Journal of Pattern Recognition and Artificial Intelligence 28 (1), 1450001. 2015.
4. Vento, M., "A Long Trip in the Charming World of Graphs for Pattern Recognition". Pattern Recognition 48 (2), pp. 291–301 2015.
5. Kashif, M., Deserno, T. M., Haak, D., Jonas, S., "Feature Description with SIFT, SURF, BRIEF, BRISK, or FREAK? A General Question Answered for Bone Age Assessment". Comput. Biol. Med. 68, pp. 67–75, 2016.
6. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., "SURF: Speeded up Robust Features". Computer Vision and Image Understanding 110 (3), pp. 346–359, 2008.
7. Lowe, D.G., "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 60 (2), pp. 91–110. 2004.
8. Gold, S., Rangarajan, A., "A Graduated Assignment Algorithm for Graph Matching". Pattern Analysis and Machine Intelligence 18 (4), pp. 377–388. 1996.
9. Riesen, K., Bunke, H., "Approximate Graph Edit Distance Computation by Means of Bipartite Graph Matching". Image Vision Comput. 27 (7), pp. 950–959. 2009.
10. Serratosa, F., "Fast Computation of Bipartite Graph Matching". Pattern Recognition Letters 45, pp. 244–250. 2014.
11. Serratosa, F., "Speeding up Fast Bipartite Graph Matching through a New Cost Matrix". Internal Journal of Pattern Recognition & Artificial Intelligence 29 (2), 1550010. 2015.
12. Serratosa, F., Alquézar, R., Sanfeliu, A., "Function-Described Graphs for Modelling Objects Represented by Attributed Graphs". Pattern Recognition 36 (3), pp. 781–798, 2003.
13. Sanfeliu, A., Serratosa, F., Alquézar, R., "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.
14. Ferrer, M., Valveny, E., Serratosa, F., "Median Graph: A New Exact Algorithm using a Distance Based on the Maximum Common Subgraph". Pattern Recognition Letters 30 (5), pp. 579–588. 2009.
15. Ferrer, M., Valveny, E., Serratosa, F., Riesen, K., Bunke, H., "Generalized Median Graph Computation by Means of Graph Embedding in Vector Spaces". Pattern Recognition 43 (4), pp. 1642–1655, 2010.
16. Kuhn, H.W., "The Hungarian Method for the Assignment Problem Export". Naval Research Logistics Quarterly 2 (1-2), pp. 83–97. 1955.
17. Jonker, R., Volgenant, T., "Improving the Hungarian Assignment Algorithm". Operations Research Letters 5 (4), pp. 171–175. 1986.
18. Sanfeliu, A., Fu, K.S., "A Distance Measure between Attributed Relational Graphs for Pattern Recognition". IEEE Transactions on Systems, Man, and Cybernetics, 13 (3), pp. 353–362. 1983.
19. Gao, X., Xiao, B., Tao, D., Li, X., "A Survey of Graph Edit Distance". Pattern Analysis and Applications 13 (1), pp. 113–129. 2010.
20. Solé, A., Serratosa, F., Sanfeliu, A., "On the Graph Edit Distance Cost: Properties and Applications". Intern. Journal of Pattern Recognition and Artificial Intelligence 26 (5), 1260004. 2012.
21. Serratosa, F. "Computation of Graph Edit Distance: Reasoning about Optimality and Speed-up". Image and Vision Computing 40, pp. 38-48, 2015.
22. Timoftea, R., Goola, L.V., "Adaptive and Weighted Collaborative Representations for Image Classification". Pattern Recognition Letters, 43 (1), pp. 127–135. 2014.
23. Saha, S., Ekbal, A., "Combining Multiple Classifiers using Vote Based Classifier Ensemble Technique for Named Entity Recognition". Data & Knowledge Engineering 85, pp. 15–39. 2013.
24. Serratosa, F., Cortés, X., Solé, A., "Component Retrieval Based on a Database of Graphs for Hand-Written Electronic-Scheme Digitalisation". Expert Systems with Applications 40, pp. 2493–2502. 2013.
25. Moreno-García, C. F., Serratosa, F., "Consensus of Two Sets of Correspondences through Optimisation Functions". Pattern Analysis and Applications, pp.1–13, 2015.
26. Moreno-García, C. F., Serratosa, F., "Weighted Mean Assignment of a Pair of Correspondences using Optimisation Functions". Syntactic and Structural Pattern Recognition LNCS 8621, pp. 301–311, 2014.
27. Moreno-García, C.F., Serratosa, F., "Consensus of Multiple Correspondences between Sets of Elements". Computer Vision and Image Understanding 142, pp. 50–64. 2015.
28. Moreno-García, C.F., Serratosa, F., Cortés, X., "Iterative Versus Voting Method to Reach Consensus Given Multiple Correspondences of Two Sets", Iberian Conference on Pattern Recognition and Image Analysis LNCS 9117, pp. 530–540, 2015.
29. Moreno-García, C. F., Serratosa, F., Cortés, X., "Consensus of Two Graph Correspondences through a Generalization of the Bipartite Graph Matching". Graph Based Representations and Pattern Recognition LNCS 9069, pp. 87–97, 2016.
30. Moreno-García, C.F. Serratosa, F., "Online Learning the Consensus of Multiple Correspondences between Sets," Knowledge-Based Systems 90, pp. 49–57, 2015.
31. Riesen, K., Bunke, H., Fischer, A., "Improving Graph Edit Distance Approximation by Centrality Measures". International Conference on Pattern Recognition, pp. 3910–3914. 2014.
32. Serratosa, F., Cortés, X., "Graph Edit Distance: Moving from Global to Local Structure to Solve the Graph-matching Problem". Pattern Recognition Letters 65 (1), pp. 204-210. 2015.
33. Papadimitriou, C., Steiglitz, K., "Combinatorial Optimization: Algorithms and Complexity". Dover Publications. 1998.
34. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E., "Convergence Properties of the Nelder-mead Simplex Method in Low Dimensions". SIAM Journal of Optimization 9 (1), pp. 112–147, 1998.
35. Ghahramani, Z., "Unsupervised Learning. Advanced Lectures on Machine Learning", pp. 72–112. Springer-Verlag. 2004.
36. Serratosa, F., Sanfeliu, A., "Signature versus Histograms: Definitions, Distances and Algorithms". Pattern Recognition 39 (5), pp. 921–934. 2006.
37. http://deim.urv.cat/~francesc.serratosa/databases/
38. http://www.featurespace.org
39. Rosten, E., Reid Porter, R., Drummond, T., "Faster and Better: A Machine Learning Approach to Corner Detection". IEEE Trans. Pattern Analysis and Machine Intelligence 32, pp. 105–119, 2010.
40. Harris, C., Stephens, M., "A Combined Corner and Edge Detector". Proceedings of the 4th Alvey Vision Conference, pp. 147–151, 1988.
41. Shi, J., Tomasi, C., "Good Features to Track". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600. 1994.
42. http://es.mathworks.com/help/vision/ref/matchfeatures.html