Error-tolerant coarse-to-fine matching model for hierarchical graphs

Pau Riba, Josep Lladós, and Alicia Fornés

Computer Vision Center - Computer Science Department
Universitat Autònoma de Barcelona, Spain
{priba,josep,afornes}@cvc.uab.es
http://www.cvc.uab.es

Abstract. Graph-based representations are effective tools to capture structural information from visual elements. However, retrieving a query graph from a large database of graphs implies a high computational complexity. Moreover, these representations are very sensitive to noise or small changes. In this work, a novel hierarchical graph representation is designed. Using graph clustering techniques adapted from graph-based social media analysis, we propose to generate a hierarchy able to deal with different levels of abstraction while keeping information about the topology. For the proposed representations, a coarse-to-fine matching method is defined. These approaches are validated using real scenarios such as classification of colour images and handwritten word spotting.

Keywords: Graph matching, Hierarchical graph, Graph-based representation, Coarse-to-fine matching

1 Introduction

Graph-based representations play an important role in content-based image retrieval. Using graphs, not only statistical information is codified but also the relations between the compounding parts. The use of graph representations in computer vision has two main requirements. First, the extraction of the structures underlying the visual objects. Second, error-tolerant metrics coping with noise or distortion must be designed. Graph matching is one of the most important challenges of graph processing [6]. Generally speaking, the problem consists in finding the best correspondence between the sets of vertices of two graphs preserving the underlying structures. The intrinsic variability of patterns, noise and errors produced from the graph extraction process, makes mandatory to encode tolerance to errors into graph matching frameworks. Thus error-tolerant graph matching has to be applied.

Graph edit distance [9] is the process of evaluating the similarity of two different graphs computing the minimum edit cost from the source to the target graph in terms of node and edge insertion, deletion and substitution. It is an optimal method and the computational complexity is exponential in the number

of nodes. A suboptimal approximation called *bipartite graph matching* was proposed by Riesen *et al.* [16]. It is based on the assignment problem solution using a cost matrix which codifies the edit operations costs. More recently, an efficient approach was proposed by Zhou and De la Torre [20] formulating graph matching as a quadratic assignment. To avoid the computation of the large pairwise affinity matrix, they propose a factorisation into smaller matrices that encode the local structure of each graph and the pairwise affinity between edges.

When dealing with large scale data, indexation strategies are required to prune the number of graph comparisons. Generally, graph indexing is solved by graph factorisation techniques where the dataset of graphs is decomposed in smaller ones representing a codebook of compounding structures. The indexation is formulated in terms of matching the constituent graphs organised in a look-up table structure. Usually, path-based methods are used to split the graphs into small redundant fragments. *GraphGrep* [17] enumerates all the existing paths up to a predefined length. This reduces the search space performing the exact matching using only few graphs. A relevant work was proposed by Yan *et al.* [19]. They propose to use frequent substructures instead of path-based methods as indexing features. Frequent graph substructures are obtained by graph sequentialization, according to a *depth first search* (DFS) traversal of the graph edges. Edge sequences are organised in a prefix tree called the *gIndex* tree. Riba *et al.* [15] proposed a binary embedding for the local context of each node. A vote scheme is used for indexation, so the subgraphs with more votes are more accurately analyzed in a finer matching process.

The above methods rely on local structures rather than global knowledge of the graph. An interesting alternative is to use a scale-space approach where the input data is hierarchically organized, summarizing it in order to avoid complex graph comparisons. Several hierarchical graph approaches have been proposed. Brun and Kropatsch [4] introduces a set of relationships between regions of a partition through irregular graph pyramids. Broelemann *et al.* [3] propose to deal with noise such as spurious nodes and edges through a hierarchical representation of plausible graphs. Ahuja and Todorovic [1] present a region based approach for object recognition based in multi-scale region segmentation. Conte *et al.* [5] propose a similar graph multi-resolution approach in order to improve the object tracking in a video. Mousavi *et al.* [11] use a hierarchical graph representation in order to improve the information codified by graph embedding frameworks. Indexation frameworks have been also proposed.

The main contribution of this work is a hierarchical graph representation and matching able to discard non-promising structures. Our hierarchical information avoids a direct matching at the original graph. The hierarchy is designed to perform a big reduction of the graphs drastically reducing the matching time. The proposed approaches are validated using real scenarios such as classification of colour images and handwritten word spotting. In the next sections we describe the representation, the matching and the results respectively.

2 Hierarchical Attributed Graph Representation

2.1 Hierarchy construction

A hierarchical graph representing information at different levels of abstraction (contraction) allows to perform the retrieval problem in an abstract manner.

Definition 1 (Hierarchical Graph). *A* hierarchical graph H *is defined as a 6-tuple* $H(V, E_N, E_H, L_V, L_{E_N}, L_{E_H})$ *where* V *is the set of nodes;* $E_N \subseteq V \times V$ *are the neighborhood edges;* $E_H \subseteq V \times V$ *are the hierarchical edges;* $\mathrm{L_V}$, $\mathrm{L_{E_N}}$ *and* $\mathrm{L_{E_H}}$ *are three labeling functions defined as* $\mathrm{L_V} : V \to \Sigma_V \times A_V^k$, $\mathrm{L_{E_N}} : E_N \to \Sigma_{E_N} \times A_{E_N}^l$ *and* $\mathrm{L_{E_H}} : E_H \to \Sigma_{E_H} \times A_{E_H}^m$, *where* Σ_V, Σ_{E_N} *and* Σ_{E_H} *are three sets of symbolic labels for vertices and edges,* A_V, A_{E_N} *and* A_{E_H} *are three sets of attributes for vertices and edges, respectively, and* $k, l, m \in \mathbb{N}$.

Given a graph G, two functions are needed to construct a hierarchical graph H:

- *Contraction:* $c : G \to H$, defines the groups of nodes that are gathered together. The contraction process can follow different criteria such as topology, features of the nodes or edges, etc. This function follows a clustering process.
- *Embedding:* $\varphi : G \to \mathbb{R}^n$, returns a vectorial representation of the contracted subgraph to be used as an attribute. The embedding function can be seen as a signature of the subgraph that summarizes the information from one level to another (information propagation between levels).

We propose a contraction criterion based on the topology. The embedding function is applied to all contracted groups of nodes propagating the information. This function is application dependent and is specified for each particular case.

2.2 Hierarchy construction by community detection

To determine the group of nodes that are joined into a unique vertex, the Girvan-Newman algorithm [10] is applied. This is a well-known method for for community detection in complex systems with complexity $\mathcal{O}(m^2 n)$, where m and n are the number of edges and nodes respectively. It is a global divisive algorithm which removes the appropriate edge at each step until all the edges are deleted. The *betweenness centrality* measure is used as edge selection. The *betweenness centrality* of $e \in E$ is defined as the number of shortest walks between any pair of nodes that cross e. The idea is that the edges with higher centrality are candidates to connect two clusters. After the edge deletion, each connected component is considered as a cluster in the hierarchy. This algorithm consists of 4 steps:

1. Calculate the betweenness centrality (BC) for all edges in the network.
2. Remove the edge with highest BC and generate a cluster for each connected component.
3. Recalculate BCs for all edges affected by the removal.
4. Repeat from step 2 until no edges remain.

The output of this algorithm is a dendrogram providing a hierarchical clustering of the graph nodes. In case of ties (i.e. several edges have the same BC), The edge with more connections in their compounding nodes is deleted. From it, we contract clusters containing at least two nodes. Moreover, it does not allow any node to be a cluster individually. Therefore, the reduction ratio is at least of 2. Afterwards, the corresponding nodes are contracted into only one vertex which is labelled with the embedding function applied to these subgraphs. The idea is that each node of the hierarchy represents a subgraph and provide information about its topology. Finally, connected communities will create connected nodes.

2.3 Splitting of articulation points

There are cases where slight deformations in the input graphs can lead to completely different hierarchies. Figure 1 shows a common subgraph that can lead to two possible hierarchies. This ambiguity can result in matching errors. Although *overlapping community detection* techniques have been developed [13], they generate redundant information leading to a bad abstraction. This problem usually comes from a symmetry in the original graph. To tackle with this problem we define *articulation points* as follows:

Definition 2 (Articulation Point). *A node in an undirected graph is an articulation point if and only if removing it the number of connected components of the graph increases.*

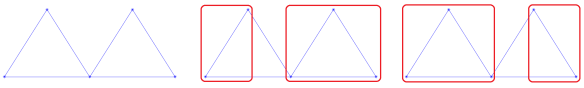

Fig. 1. Ambiguity configuration that can significantly influence in the hierarchy construction, in red two possible clusterings of nodes from the contraction function.

These nodes are of key importance, if they are classified in an incorrect cluster, they can change significantly the topology. Thus, we propose to split the articulation points of the graphs creating *virtual* nodes and disconnecting them. Hence, the hierarchical representation is stabilised without introducing noise to the data. The articulation points therefore divide and belong to two or more clusters. Introducing this modification to the contraction function, a more stable hierarchy is generated. Figure 2 shows the splitting process in a real scenario where graphs represent skeleton features in handwritten word images.

3 Error tolerant hierarchical matching

As graph matching baseline, we have used the algorithm of *bipartite graph matching* proposed by Riesen and Bunke in [16]. It uses a cost matrix that codifies the

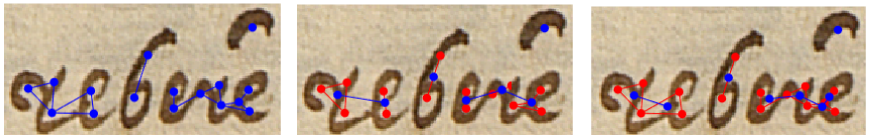

Fig. 2. From left to right: input graph, hierarchy for the proposed contraction function and hierarchy splitting the articulation points. In red, the contracted nodes.

edit costs between the source and target nodes. Once the cost matrix is defined, an edit operation is assigned to each node minimising the total cost. We have used the same edit costs as [14]: node substitution cost is based in the distance, the attributes and local structure of incident edges; edge substitution cost is computed in terms of edge attribute, angle and length; predefined costs are defined for node and edge insertion and deletion. Thus, there are 8 parameters: 3 (node substitution), 3 (edge substitution) and 2 (insertion and deletion).

To take advantage of the hierarchical representation, we propose a coarse-to-fine graph matching approach. Let us denote H^i the graph representation at level $i = 1, \ldots, N$. It iteratively refines the matching starting at the coarsest level (i.e. $i = N$). The comparison is performed using bipartite graph matching taking the graph representation at level i without the hierarchical edges. If the distance at level i is small enough, the matching is performed at the next level $(i - 1)$. The threshold to decide whether to advance in the hierarchy or not is application dependent and a threshold is set experimentally. Starting the matching at the abstract level avoids a high number of comparisons at more detailed levels where the graphs are significantly bigger. Ideally, the last level is only used for graphs that are very similar to the input one. The information about the matching level is kept. Figure 3 shows the iterative process to decide whether the graphs match or we can discard the comparisons in any of the abstract levels of the hierarchy.

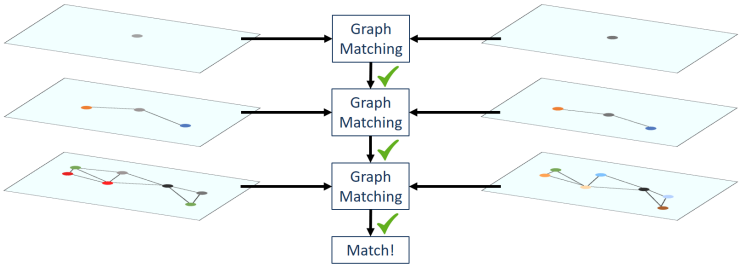

Fig. 3. Coarse-to-fine matching scheme.

4 Experiments

4.1 Datasets

Different databases have been used. First, the *Columbia Object Image Library* (COIL-100) [12] and the *Object DataBank* (ODBK) [18] have been used to reproduce the experiments proposed by Mousavi *et al.* [11] in an object classification scenario. Second, the *Barcelona Historical Handwritten Marriages* (BH2M) database has been used in a graph-based word spotting scenario in handwritten documents where graphs are irregular and suffer from high distortions.

The COIL database consists of images of 100 different objects taken at 72 equally spaced poses whereas the ODBK database is formed by 209 3D objects with 14 views. Graph nodes are extracted using the *Harris corner detector*. The edges are generated using the Delaunay triangulation on these nodes. The final graphs are not weighted for the edges and store the coordinates for the nodes. For the experimentation, 15 and 50 classes, with maximum average number of nodes are used. The graphs are divided into three sets, training, validation and test of 360, 75 and 150 for the COIL dataset and 300, 150 and 150 for the ODBK database. Figure 4 shows some examples coming from these databases.

Fig. 4. Example of objects from the COIL-100 and ODBK databases.

The BH2M database [8] corresponds to marriage licenses written between 1617 and 1619. It contains 174 handwritten pages divided into training (100), validation (34) and test (40). The handwritten words are represented by attributed graphs where nodes correspond to basic primitives called *graphemes* [14]. Graphemes defined as convexities are described using the *Blurred Shape Model* (BSM) descriptor [7]. The descriptors extracted from the training set are used to create a *codebook*, from which node labels are set. Edges represent adjacency relations between those primitives. Figure 2 shows an example of the obtained graphs plotted on the image.

4.2 Results

The experiments have been divided into two challenges: *object classification* and *word spotting*. Thresholds have been carefully selected using the validation set.

Object Classification: The use of a richer representation allows us to use a simple classification approach (k-NN) achieving similar results than an scheme with a less expressive representation and complex classifier, with the advantage of reducing the computational cost. The selected embedding function encodes

information of the Morgan Index of length 1 and 2 of the previous level. Three approaches have been evaluated for the information selection: averaging the Morgan Index and node position from all contracted nodes, averaging the Morgan Index and selecting the most connected node, and taking the maximum Morgan Index and the most connected node position.

Each level of the hierarchy has been validated alone and combined with the original graph to explore the benefits of the proposed coarse-to-fine matching. All the parameters for the distance computation have been chosen performing a random search in the validation set. Since the graphs are generated using a triangulation, there are not articulation points, therefore, both contraction functions will lead to the same hierarchy.

Table 1 shows the performance for COIL and ODBK databases respectively. For this experiment, the mean of the node positions and the Morgan Index is used as embedding function. The last 3 rows correspond to the performance reported by [11] using their hierarchical representation with the same graphs.

Table 1. Performance for Object Classification for COIL (left) and ODBK (right) datasets. Rows are divided in 5 blocks: performance for each level; coarse-to-fine matching using the 1st, 2on abstract levels and the combination of them; the final row-block correspond the the performance reported by Mousavi *et al.*. Columns correspond to the used threshold; accuracy of a k-NN classifier; percentatge of avoided comparisons at the base level; time in seconds to perform all the comparisons.

COIL database						
	Thresh.	K-NN(%) 1	3	5	AC[1](%)	t(s)
Original	-	100.00	100.00	98.00	-	2010
1st abst.	-	72.67	74.67	72.67	-	167
2nd abst.	-	38.00	39.33	44.67	-	13
1st abst.	0.1982	98.00	97.33	93.33	67.37	977
	0.1680	90.00	89.33	82.67	95.41	289
2nd abst.	0.2153	100.00	99.33	96.67	33.68	1444
	0.1895	97.33	94.67	93.33	58.99	937
1st abst. 2nd abst.	0.1982 0.2153	98.67	98.00	92.67	71.63	893
Original	-	100.00	97.00	90.00	Mousavi *et al.* [11]	
1st abst.	-	98.17	94.83	88.83		
2nd abst.	-	87.00	81.67	78.17		

ODBK database						
	Thresh.	K-NN(%) 1	3	5	AC[1](%)	t(s)
Original	-	79.33	76.00	74.00	-	34959
1st abst.	-	58.67	58.00	54.67	-	1954
2nd abst.	-	42.00	41.33	46.00	-	141
1st abst.	0.2396	79.33	76.00	74.00	48.18	22501
	0.2130	78.67	75.33	72.00	79.10	10496
2nd abst.	0.2973	78.67	74.67	72.67	33.76	26111
	0.2573	76.67	71.33	68.67	68.49	12228
1st abst. 2nd abst.	0.2130 0.2973	78.00	74.00	70.67	79.23	10292
Original	-	66.67	65.33	63.33	Mousavi *et al.* [11]	
1st abst.	-	66.67	62.67	62.00		
2nd abst.	-	60.00	55.33	53.33		

Note that the big loss of performance between the abstract levels is corrected choosing a good trade-off between them. We are able to prune more than the 50% of comparisons at the finest level while achieving good results. For instance, choosing a conservative threshold, the time reduction is half, losing only 2% of accuracy for the COIL database and 1.5 times faster maintaining the same accuracy for the ODBK database. However, relaxing this threshold, we are able to achieve a speed-up of 7x with a loss of 10% in accuracy for the COIL database

and 3.3 times faster losing 1% in accuracy for ODBK. In a large scale scenario, this is an acceptable loss to make an application much faster. Compared to [11], our methodology do not achieve as good results as them in the different abstraction levels. One of the main reasons is that our hierarchy is dynamically constructed, not fixing the contraction degree and generating smaller graphs.

Word Spotting: Word spotting is the task of retrieving word images from a document image similar to a given query (text or image). It is formulated as a visual object detection problem. Most word spotting techniques use statistical representations (e.g. HOG, SIFT) of the word images, e.g. [2]. The *embedding function* consists in a vector that counts the number of paths of length up to k from any node to a node with label i. The best configuration has been $k = 0$, i.e. counting the number of nodes with label i (similar to a *bag of words* for the nodes). As a retrieval problem the *mean average precision* (mAP) has been used for the evaluation. Using the same parameters proposed in [14] in order to compute the edit cost operations a mAP of 69.45% is achieved. Reproducing the same experiment using the **first level** of the hierarchy the achieved mAP is 35.67%. By **splitting the articulation points** as proposed in Section 2.3 achieves a mAP of 46.37%. Figure 5 shows the interpretation of the hierarchical graph representation in the context of this database. Observe how the graphemes are combined at each level to create more complex shapes like letters, bi-grams and finally words.

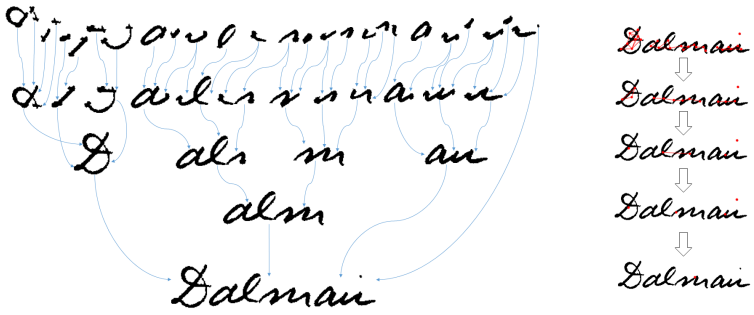

Fig. 5. Hierarchy construction for the word "Dalmau".

Table 2 shows a comparison between: the original graphs, the proposed framework with two threshods, and graph indexation [15]. Recall (R) and Specificity (SPC) are computed on the selected graphs using the first abstract level as classifier. Notice that the proposed hierarchical framework achieves high specificity whereas keeping a better trade-off with the recall than the indexation approach. Moreover, only losing 8% of mAP which is acceptable for a large scale retrieval we are able to speed up the process almost 5 times.

[1] AC stands for Avoided Comparison

Table 2. Comparison of the proposed hierarchical framework against an indexation framework [15]. The mean average precision corresponds to the evaluation of the word spotting problem; recall (R) and specificity (SPC) are computed on the selected graphs using the hierarchy or the indexation respectively; finally, time per query is provided.

	mAP (%)	R (%)	SPC (%)	Time/query[2] (s)
Original	69.45	100.00	0.00	19.58
+abst. (t=0.30)	**68.27**	90.91	69.98	**12.46**
+abst. (t=0.25)	**61.71**	67.93	97.91	**3.94**
+[15] (t=0.20)	66.13	92.54	46.13	16.34
+[15] (t=0.30)	61.15	83.55	63.04	12.74

5 Conclusions

This paper has presented a construction of a hierarchical graph representation by means of contraction and embedding functions. Contraction uses graph clustering techniques to gather nodes and simplify the graph. Moreover, a modification of the contraction function has been proposed to stabilise the hierarchy in certain graphs. The proposed method is able to significantly reduce the graph size allowing a fast graph comparison through a coarse-to-fine matching approach. This methodology prunes the amount of comparisons in the fine level. The approach has been exhaustively validated using several databases for of large-scale graph retrieval. Compared to other related works, the proposed approach dynamically gathers the nodes without predefining the number of clusters, therefore, the ratio of reduction for each sample can change. Furthermore, the graph size is extremely reduced from one level to another.

We conclude that hierarchical graph representations are a powerful tool in the matching process. This representation gives information about the relation of a group of nodes (those that are contracted) instead of the typical pair-wise relations. Moreover, each level of the hierarchy can be enriched following other indexation methodologies such as [15]. The future work will be focused on the development of matching algorithms using the whole representation at once.

Acknowledgments

This work has been partially supported by the Spanish project TIN2015-70924-C2-2-R, a FPU fellowship FPU15/06264 from the Spanish Ministerio de Educación, Cultura y Deporte, the Ramon y Cajal Fellowship RYC-2014-16831 and the CERCA Programme/Generalitat de Catalunya.

References

1. N. Ahuja and S. Todorovic. From region based image representation to object discovery and recognition. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 1–19. Springer, 2010.

[2] 1000 queries selected randomly against 13098 graphs

2. J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566, Dec 2014.
3. K. Broelemann, A. Dutta, X. Jiang, and J. Lladós. *Hierarchical Plausibility-Graphs for Symbol Spotting in Graphical Documents*, pages 25–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
4. Luc Brun and Walter Kropatsch. Contains and inside relationships within combinatorial pyramids. *Pattern Recognition*, 39(4):515 – 526, 2006. Graph-based Representations.
5. D. Conte, P. Foggia, J.M. Jolion, and M. Vento. A graph-based, multi-resolution algorithm for tracking objects in presence of occlusions. *Pattern Recognition*, 39(4):562 – 572, 2006. Graph-based Representations.
6. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.
7. S. Escalera, A. Fornés, O. Pujol, P. Radeva, G. Sánchez, and J. Lladós. Blurred shape model for binary and grey-level symbol recognition. *Pattern Recognition Letters*, 30(15):1424 – 1433, 2009.
8. D. Fernández-Mota, J. Almazán, N. Cirera, A. Fornés, and J. Lladós. Bh2m: The barcelona historical handwritten marriages database. *International Conference on Pattern Recognition*, 2014.
9. X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.
10. M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *National Academy of Sciences*, 99(12):7821–7826, 2002.
11. S.F. Mousavi, M. Safayani, A. Mirzaei, and H. Bahonar. Hierarchical graph embedding in vector space by graph pyramid. *Pattern Recognition*, 61:245–254, 2017.
12. S. Nayar, S. Nene, and H. Murase. Columbia object image library (coil 100). *Dept. of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96*, 1996.
13. G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.
14. P. Riba, A. Fornés, and J. Lladós. Handwritten word spotting by inexact matching of grapheme graphs. In *13th International Conference on Document Analysis and Recognition*, pages 781–785, Aug 2015.
15. P. Riba, J. Lladós, A. Fornés, and A. Dutta. Large-scale graph indexing using binary embeddings of node contexts for information spotting in document image databases. *Pattern Recognition Letters*, pages –, 2016.
16. K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009.
17. D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the 21st Symposium on Principles of Database Systems*, pages 39–52, New York, USA, 2002. ACM.
18. M. J. Tarr. The object databank, 2011.
19. X. Yan, P.S. Yu, and J. Han. Graph indexing: a frequent structure-based approach. In *Proceedings of the int. conference on Management of data*, pages 335–346, 2004.
20. F. Zhou and F. de la Torre. Factorized graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2015.