Graph Matching Framework

Ekaterina Tikhoncheva

1 Problem statement

Consider two undirected weighted graphs $G^I = (V^I, E^I, D^I)$ and $G^J = (V^J, E^J, D^J)$, where V, E, D denote set of nodes, set of edges and set of node attributes respectively. We assume situation, where $|V^I| = n_1$, $|V^J| = n_2$ and n_1 is not necessary equal to n_2.
The aim of graph matching is to find a subset of possible node correspondences, which maximizes the similarity value between two graphs. Such subset can be represented by a binary vector $x \in \{0,1\}^{n_1 n_2}$, where $x_{(j-1)n_1+i} = 1$, if node $v_i \in V^I$ is matched to node $u_j \in V^J$, and $x_{(j-1)n_1+i} = 0$ otherwise. For simplicity we will write further x_{ij} instead of $x_{(j-1)n_1+i}$.
To measure similarity between graphs we define two similarity functions: *nodes similarity function* (first-order similarity) $s_V(v_i, u_j),\ v_i \in V^I, u_j \in V^J$ and *edge similarity function* (second-order similarity) $s_E(e_{ii\prime}, e_{jj\prime}),\ e_{ii\prime} \in E^I, e_{jj\prime} \in E^J$. Both functions can be combined in one *similarity* or *affinity matrix* $S \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$, whose diagonal elements are $s_V(v_i, u_j)$ and non-diagonal elements are $s_E(e_{ii\prime}, e_{jj\prime})$.
Using this notation one can formulate *one-to-one Graph Matching Problem* **GMP** as an quadratic optimization problem [2, 3, 4, 5]:

$$\begin{aligned} &\underset{x}{\operatorname{argmin}}\, x^T S x \\ \text{s.t. } &x \in \{0,1\}^{n_1 n_2} && (1)\\ &\sum_{i=1\ldots n_1} x_{ij} = 1 && (2)\\ &\sum_{j=1\ldots n_2} x_{ij} = 1 && (3) \end{aligned}$$

The maximum number of possible matches is equal to $\min(n_1, n_2)$. That means, in case when $n_1 \neq n_2$, only one of the conditions (2) or (3) will be fulfilled.
Quadratic Optimization Problem is known to be *NP*-hard [12]. This limits greatly the size of a graph, for which a exact solution can be calculated in reasonable time. Due to this there is a number of algorithms () that solve graph matching problem inexact.
Unfortunately, most of the algorithms have two following problems:

1. they are still limited in size of permissible graphs. Experiments in most of the papers consider graphs with up to 100 nodes.

2. possible presence of outliers can reduce the accuracy of matching algorithm [13].

Our main aim was to develop a framework, which would allow an existing graph matching algorithm to cope with both problems.

2 Approach

The main idea of our approach is to create flexible multi-scale matching framework, that could improve the speed of theoretically any matching algorithm for large graphs and increase their accuracy.
Given an initial graphs G^I and G^J we create for each of them a coarse representative graphs A^I, A^J. Each node of such graph represent a subgraph of initial graph. We refer further to nodes of the coarse graphs A^I, A^J as *anchor nodes* or just *anchors* and to the graphs themself as *anchor graphs*.
Initial graphs with corresponding anchor graphs build two level structure (see fig ref-fig:2levels): initial graphs (or *fine graphs*) represent a lower level and anchor graphs represent a higher level.
In case of large G^I and G^J graphs, where existing inexact solution techniques are difficult to apply due to time and storage complexity, their anchor graphs can be constructed small enough to not have this problems. That makes it possible, to find a solution of GMP on the higher level fast.
Ones the matching problem was solved on the higher level, we get the correspondences between subgraphs of the initial graphs, which are also much smaller comparing to complete graphs. Solving after that GMP for pairs of subgraphs lead us to the desirable correspondences between nodes of fine graphs.
Obviously, accuracy of such two level matching approach depends heavily on the partition of initial graphs into subgraphs and on the matching quality of anchor graphs. To make the approach robuster we integrate the result of matching on the lower level into graph partition algorithm and repeat the whole scheme several times till convergence.
In our work we concentrate ourself on the task of finding feature correspondences between two images. In the next sections we describe detailed the construction of initial graphs from given images and corresponding anchor graphs together with the matching algorithm on both levels and propagation of the matching results between the levels.

2.1 Lower Level Graph Construction

In this section we describe structure of a lower level graphs built for given images I and J.

Features, that was collected densely on the edges [6], represent the sets of nodes V^I, V^J of the graphs G^I, G^J respectively. This results in hundreds or thousand of nodes. As node attributes D^I, D^J we used *SIFT descriptors* [10] with fixed orientation and scale. The nodes of the graphs are connected with edges with their k nearest neighbors.
To calculate affinity matrix S between two lower level graphs G^I, G^J or their subgraphs we need to define similarity functions between nodes and edges. Similarity of two nodes $v_i \in V^I, u_j \in V^J$ is equal to cosine similarity of their descriptors. For the pairwise edge similarity we used the same formula as in [2, 13], i.e.

$$s_E(e_{ii\prime}, e_{jj\prime}) = exp(-\frac{(l_{ii\prime} - l_{jj\prime})^2}{\sigma_s^2}) \tag{4}$$

where $l_{ii\prime}$, $l_{jj\prime}$ are the lengths of edges $e_{ii\prime} \in E^I$ and $e_{jj\prime} \in E^J$ respectively.

2.2 Higher Level Graph Construction

We define an anchor graph of a given fine graph $G = (V, E, D)$ as $A = (V^A, E^A, U^A)$, where V^A is set of anchors, E^A is set of edges between the anchors and U^A is correspondence matrix between anchors and nodes of G.
Each anchor $a_k \in V^A$ represent a subgraph $G_k = (V_k, E_k, D_k)$ of G. In case of m anchors and n nodes in V matrix U^A is $n \times m$ binary matrix with

$$U^A_{ik} = \begin{cases} 1, & \text{if node } v_i \in V_k \\ 0, & \text{otherwise} \end{cases}$$

2.2.1 Construction of anchor graph

To construct an anchor graph A with fixed number m of anchors from a given fine graph G we adopted coarsening phase from multi-level graph partition algorithms [1, 7, 8, 11]. Such algorithms have three phases:

1. graph coarsening phase, where one creates a hierarchy of graphs by successive merging of nodes in graph on previous stage starting with initial graph;

2. graph partition stage, where the partition problem is solved exact on the coarsest level;

3. refinement phase, where solution of the coarsest level is interpolated through all levels of the hierarchy until initial graph.

There are several types of graph coarsening algorithms. In our work we used so-called strict aggregation scheme (**SAG**) [1], which groups nodes of G in *disjoint* subsets based on the strength of the edges between them.
We implemented two SAG based algorithms: Heavy Edge Matching (**HEM**) and Light Edge Matching (**LEM**) [1]. Both algorithms visit nodes of the G in random order and construct a *matching* M of the graph. A selected node $v_i \in V(G) \setminus V(M)$ is

matched to a node $v\prime \in V(G) \setminus V(M)$ in M, if $s_{vv\prime} = \max_{u \in N(v)} s_{vu}$ in case of HEM and $s_{vv\prime} = \min_{u \in N(v)} s_{vu}$ in case of LEM. Here, $N(v)$ denotes the neighborhood of v, $V(M)$ the set of matched nodes and s the strength of an edge. In case of LEM $s_{ii\prime} = l_{ii\prime}$ and in case of HEM $s_{ii\prime} = exp(-\frac{l_{ij}}{\sigma_s^2})$.
The edges in M will be contracted, i.e. their endpoints will be replaced with a new node, that lies in the middle of contracted edge and is connected to all neighbor of its endpoints.
It is clear, that one iteration of HEM or LEM reduces the number of nodes in G at most by $\lfloor \frac{n}{2} \rfloor$ nodes. To get an coarse graph with m nodes the coarsening algorithm should be repeated several times.
Obtained at the end coarse graph is the desired anchor graph A of G.

2.2.2 Similarity matrix

Assume, that for each of two given lower level graphs $G^I = (V^I, E^I, D^I)$ and $G^J = (V^J, E^J, D^J)$ we built corresponding anchor graphs $A^I = (V^{AI}, E^{AI}, U^{AI})$ and $A^J = (V^{AJ}, E^{AJ}, U^{AJ})$.
We define length of edges between two anchors a_k, $a_k\prime$ as a mean of distances between nodes of the subgraphs G_k and $G_{k\prime}$. With the other words:

$$L_{kk\prime} = mean_{v_i \in G_k, v_{i\prime} \in G_{k\prime}} l_{ii\prime} \tag{5}$$

Based on this definition we use the same formula for *edge similarity* between anchors as we set it for the lower level graphs (see Eq.4):

$$s_E^A(e_{kk\prime}, e_{pp\prime}) = exp(-\frac{(L_{kk\prime} - L_{pp\prime})^2}{\sigma_s^2}) \tag{6}$$

where $L_{kk\prime}$, $L_{pp\prime}$ are the lengths of edges $e_{kk\prime} \in E^{AI}$ and $e_{pp\prime} \in E^{AJ}$ respectively.
To measure *similarity between the anchors* we define two different descriptors of an anchor a.
The first descriptor $d_1(a)$ should measure similarity of node descriptors in corresponding subgraphs. For this purpose we adopted *Bag Of Feature Model* ref: we built a common dictionary based on the descriptors in D^I and D^J and define $d_1(a)$ as a histogram of "codewords" in corresponding subgraphs.
The second descriptor $d_2(a)$ should describe the structure similarity of subgraphs. We define $d_2(a)$ as a set of histograms $\{d_2(a, v)\}$ of the nodes v in the underlying subgraph of the anchor a. The histograms of nodes have the fix number of bins b and represent a distribution of the length of the subgraph edges inside of a small circle region around each node.
The similarity value between two anchors can be calculated now based on the first or second type of anchor descriptors by calculation of a distance between histograms based on χ^2 statistic test [14]:

$$s_1(a_k, a_p) = \sum_{b_i \in B} \frac{(d_1(a_k) - d_1(a_p))^2}{(d_1(a_k) + d_1(a_p))} \tag{7}$$

$$s_2(a_k, a_p) = \frac{1}{|V(G_k)|}\frac{1}{V(G_p)|}\sum_{v\in V(G_k)}\sum_{u\in V(G_p)}\Big(\sum_{b_i\in B}\frac{(d_2(a_k,v)-d_2(a_p,u))^2}{(d_2(a_k,v)+d_2(a_p,u))}\Big) \quad (8)$$

One cal also use a combination of both anchor similarity functions.

2.3 Matching Algorithm

Generally, it is possible to use in our approach arbitrary graph matching algorithm for finding correspondences between anchors (higher level) and nodes (lower level). However we selected *Reweighted Random Walks Method* (**RRWM**) [3], because it shows high matching accuracy (64.01% in average) and is fast. Additionally, is suits well for finding common subgraph of two graphs in presence of outliers.
Important is, that the algorithm solves relaxed version of the initial problem (see section 1), where constraints (2),(3) are dropped and the integrality constrains (1) is replaced with $x \in \{0,1\}^{n_1 n_2}$. We used *greedy mapping* analog to [9], instead used by authors Hungarian algorithm for discretization of continues solutions.

2.4 Connection between two levels

3 Experimental Evaluation

References

[1] C. Chevalier and I. Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *Learning and Intelligent Optimization: Third International Conference, LION 3. Selected Papers*, pages 191–205. Springer-Verlag, 2009.

[2] M. Cho and O. Duchenne. Finding Matches in a Haystack : A Max-Pooling Strategy for Graph Matching in the Presence of Outliers. *CVPR*, 2014.

[3] M. Cho, J. Lee, and K. M. Lee. Reweighted Random Walks for Graph Matching. *ECCV*, 2010.

[4] M. Cho and K. M. Lee. Progressive graph matching: Making a move of graphs via probabilistic voting. *CVPR*, 2012.

[5] 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.

[6] P. Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

[7] B. Hendrickson and R. Leland. A Multi-Level Algorithm For Partitioning Graphs. In *Proceedings of the IEEE/ACM SC95 Conference*, pages 1–14, 1995.

[8] G. Karypis and V. Kumar. Multilevel graph partitioning schemes. In *Proc. 24th Intern. Conf. Par. Proc., III*, pages 113–122. CRC Press, 1995.

[9] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[10] D. G. Lowe. Distinctive Image Features from. *International Journal of Computer Vision*, 60:91–110, 2004.

[11] I. Safro, P. Sanders, and C. Schulz. Advanced coarsening schemes for graph partitioning. In *Experimental Algorithms*, volume 7276 of *Lecture Notes in Computer Science*, pages 369–380. Springer Berlin Heidelberg, 2012.

[12] S. Sahni. Computationally Related Problems. *SIAM J. Comput*, 3(4):262–279, 1974.

[13] Y. Suh, K. Adamczewski, and K. M. Lee. Subgraph Matching using Compactness Prior for Robust Feature Correspondence. *CVPR*, 2015.

[14] D. Van der Weken, M. Nachtegael, and E. Kerre. Some new similarity measures for histograms. In *ICVGIP*, pages 441–446, 2004.