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 the situation, where $|V^I| = n_1$, $|V^J| = n_2$ and n_1 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 a 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 a 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) $s_E(e_{ii'}, e_{jj'}),\ e_{ii'} \in E^I, e_{jj'} \in E^J$. 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 $s_E(e_{ii'}, e_{jj'})$.
Using this notation one can formulate a *one-to-one Graph Matching Problem* **GMP** as an quadratic optimization problem [2, 3, 4, 5]:

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

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.
A Quadratic Optimization Problem is known to be *NP*-hard [13]. This limits greatly the size of a graph, for which a exact solution can be calculated in reasonable time. Due to this there are a number of algorithms () that solve graph matching problem inexact. Unfortunately, most of the algorithms have the 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 (links)

2. possible presence of outliers can reduce the accuracy of matching algorithm [14].

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 a flexible multi-scale matching framework, that could improve the speed of theoretically any matching algorithm for large graphs and additionally increase their accuracy.
Given initial graphs G^I and G^J we create for each of them coarse representative graphs A^I, A^J. Each node of such a graph represents a subgraph of the 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 a two level structure (see Fig. 1): 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 these problems. That makes it possible, to find a solution of GMP on the higher level fast.
Ones the matching problem is solved on the higher level, we get the correspondences between the subgraphs of the initial graphs, which are also much smaller compared to the complete graphs. Solving after that GMP for pairs of subgraphs lead us to the desirable correspondences between nodes of fine graphs.
Obviously, the accuracy of such two level matching approach depends heavily on the partition of the initial graphs into subgraphs and on the matching quality of the anchor graphs. To make the approach more robust we integrate the result of matching on the lower level into a 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 in detail 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.

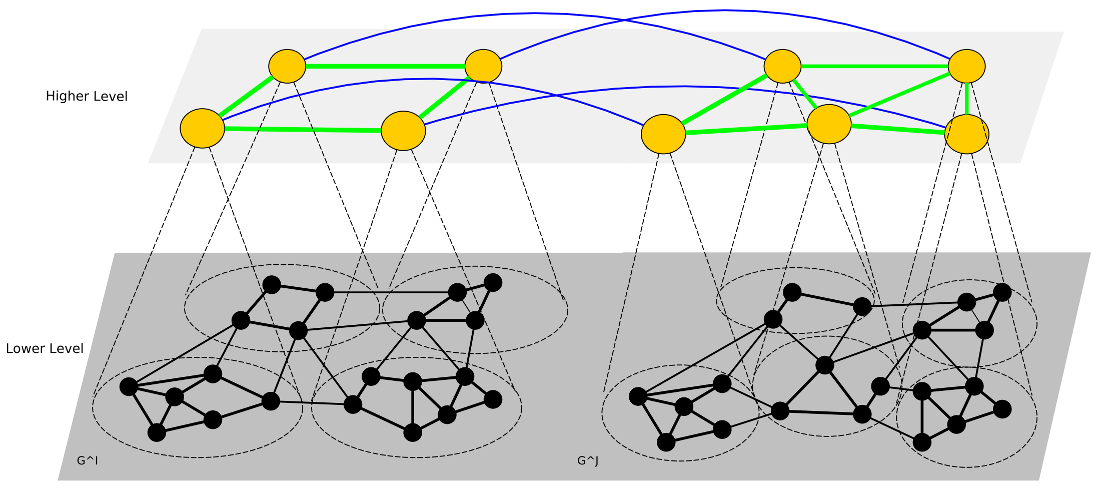


Figure 1: Two level framework for graph matching

2.1 Lower Level Graph Construction

In this section we describe the structure of lower level graphs built for given images I and J.
Features, that were 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* [11] with fixed orientation and scale. The nodes of the graphs are connected vie edges with their k nearest neighbors.
To calculate the 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, 14], i.e.

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

where $l_{ii'}$, $l_{jj'}$ are the lengths of edges $e_{ii'} \in E^I$ and $e_{jj'} \in E^J$ 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 s set of anchors, E^A is a set of edges between the anchors and U^A is a 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 a $n \times m$ 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, 8, 9, 12]. 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 the 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 a contracted edge and is connected to all neighbors 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 the corresponding anchor graphs $A^I = (V^{Ia}, E^{Ia}, U^{Ia})$ and $A^J = (V^{Ja}, E^{Ja}, U^{Ja})$.
We define the 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 other words:

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

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 $L_{kk'}$, $L_{pp'}$ are the lengths of edges $e_{kk'} \in E^{Ia}$ and $e_{pp'} \in E^{Ja}$ respectively.
To measure *similarity between the anchors* we define two different descriptors of an anchor a.
The first descriptor $d_1(a)$ should measure the 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 [15]:

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

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

One can 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], as it shows high matching accuracy (64.01% in average) and is fast. Additionally, it 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 constrain (1) is replaced with $x \in \{0, 1\}^{n_1 n_2}$. We used *greedy mapping* analog to [10], instead of Hungarian algorithm used by the authors for discretization of continues solutions.

2.4 Connection between two levels

Assume, we solved the GMP on the higher level. That means, we know pairs of correspondences between the anchor nodes: $m^a = \{(a_k, a_p)\}, a_i \in V^{Ia}, a_p \in V^{Ja}$. Since each anchor represents a subgraph of the initial graph, m^a defines at the same time the correspondences between subgraphs of the initial graphs G^I and G^J. Solution of the GMP for each pair of the subgraphs gives us the desirable correspondences between the nodes of the original graphs ($m = \{(v_i, v_j)\}, v_i \in V^I, v_j \in V^J$).

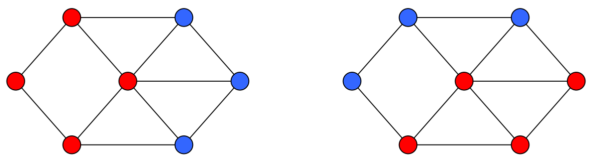


Figure 2: Example of bad partition of two equal graphs into two subgraphs

As we already mentioned above, the quality of the resulting solution m depends in our framework not only on the quality of the graph matching algorithm, but also on the graph partitioning algorithm. The Fig. 2 shows, that by existing partition the matching results will be very pure for all possible matches on the higher level.
To cope with this problem, we present an *iterative approach*, where subgraphs of the initial graphs are allowed to exchange nodes based on the solution m after each iteration of two-level graph matching algorithm. Exchanging rules are based on the affine transformation assigned to each matched pair of the subgraphs.
We consider two matched subgraphs $G_k^I = (V_k^I, E_k^I, D_k^I)$ and $G_p^J = (V_p^I, E_p^I, D_p^I)$. Based on the matching between the nodes of this pair we apply *RANSAC* [7] to estimate two *affine transformations* $T_{kp} : V_k^I \to V_p^J$ and $T_{pk} : V_p^J \to V_k^I$. Between those two transformations, we select the one with smaller transformation error. The transformation error is defined as follows. For each node $v_i \in V_k^I$ we calculate the error between its matched node $m(v_i) = v_j \in V_p^J$ and its projection $T_{kp}(v_i)$:

$$err(v_i) = \|T_k p(v_i) - m(v_i)\|_{l_2} \tag{9}$$

The error of the estimated affine transformation T_{kp} (analog for $err(T_{pk})$) is then defined as

$$err(T_{kp}) = \operatorname*{median}_{v_i \in V_k^I} err(v_i) \tag{10}$$

For the transformation with the smallest error we calculate its inverse transformation and associate both of them with the subgraph match (a_k, a_p). For simplicity we preserve the notation T_{kp} and T_{pk} for the transformations related to the subgraph match (a_k, a_p). In this way we can associate to each subgraph pair with more than 3 node correspondences [1] the estimated affine transformation of their nodes.
If the transformation law describes well the matching between the subgraphs nodes (i.e. the transformation error 10 is small), then we include nodes near $T_k p(V^I)$, $T_p k(V^J)$ in corresponding subgraph with the confidence value equal to $e^{-\min(err(T_{kp}), err(T_{pk}))}$. If one node was selected by several subgraphs, it will be included in the subgraph with the biggest associated confidence value.
If there are subgraph with less then 3 nodes, we shift this nodes to the subgraphs of their nearest neighbors. For this nodes $err(v) = M$ (see Eq. 9), where M is some big constant.
The described approach for subgraph reorganization is summarized in Algorithm 1.

[1]we need at least 3 pair of correspondences to be able to estimate an affine transformation

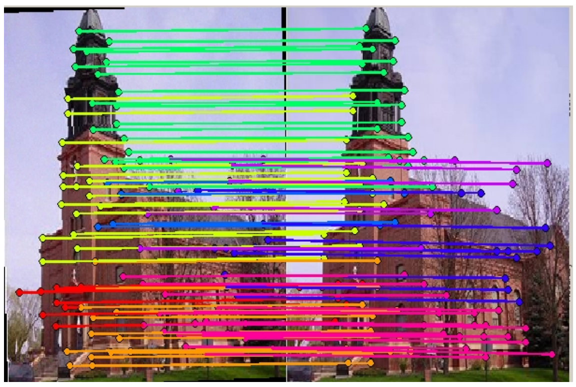
Algorithm 1: UpdateSubgraphs

```
  Input:  correspondence matrices between nodes of the initial
          graphs and anchors: $U^{Ia}$, $U^{Ja}$
          list of matches between the subgraphs: $m^a = \{(a_k, a_p)\}$;
          list of estimated affine transformations for each matched
          pair: $\{(T_{kp}, T_{pk})\}$
  Output: new correspondence matrices $\hat{U}^{Ia}$, $\hat{U}^{Ja}$
1 define subgraphs of the initial graphs based on the correspondence
  matrices $U^{Ia}$, $U^{Ja}$: $\{G^I_k\}$ and $\{G^J_p\}$
2 $\hat{U}^{Ia} = \frac{1}{2}U^{Ia}$, $\hat{U}^{Ja} = \frac{1}{2}U^{Ja}$
3 foreach matched subgraph pair $(a_k, a_p)$ do
      if $|V^I_k| \geq 3$ AND $|V^J_p| \geq 3$ then
          $err = \min(err(T_{kp}), err(T_{pk}))$ ;          // see equations 9, 10
          if $|err| < \epsilon$ then
              $\hat{U}^{Ja}(v, a_p) = \exp(-err)$ for $v \in T_{kp}(V^I_k)$
              $\hat{U}^{Ia}(v, a_k) = \exp(-err)$ for $v \in T_{pk}(V^J_p)$
      else
          shift nodes of too small subgraphs into subgraphs, their nearest neighbors
          belong to
4 set in each row of $\hat{U}^{Ia}(\hat{U}^{Ja})$ the maximum element to 1 and all other to 0
  return $\hat{U}^{Ia}$, $\hat{U}^{Ja}$
```

2.4.1 Simulated Annealing SM

After some experiments we found out, that the Algorithm 1 can stay trapped in the local optima. To cope with this we use simulated annealing.
In each iteration *after application of the Algorithm 1*, we randomly try to shift a node to one of its three nearest anchors. The energy state of the system is defined as $E = \sum_{v \in V^I} err(v)$ (see Eq. 9). The new state is calculated after one node was shifted and denoted as E^{new}. Wenn $E^{new} < E$ the shifted node stays in its new state. Otherwise, the new state will accepted with the probability $exp(-\frac{E^{new}-E}{T})$, where $T = 1/it$ is the current temperature of the system and depends on the iteration number it.
Afterwards we apply again the Algorithm 1.

3 Experimental Evaluation


(a) Without *SM*

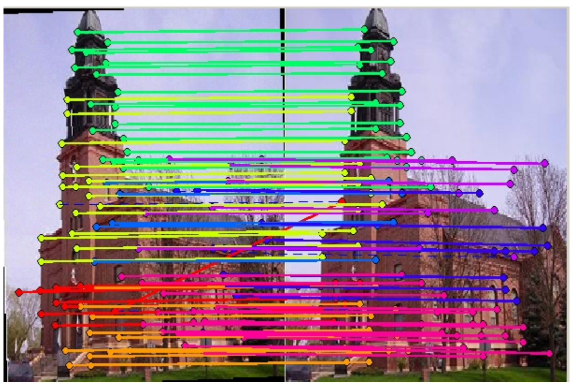
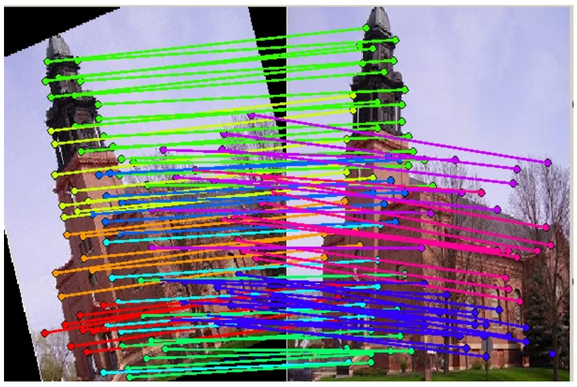

(b) With *SM*

Figure 3: Result of the two-level graph matching approach after 6 iterations


(a) Without *SM*

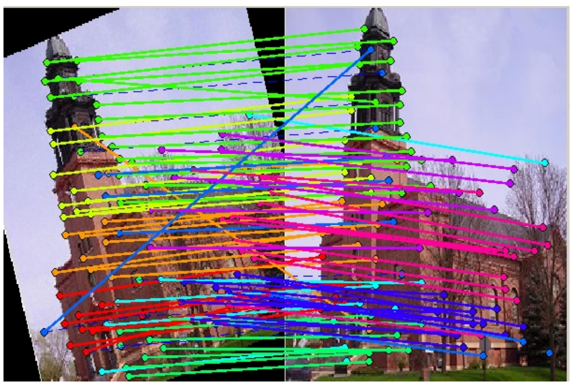
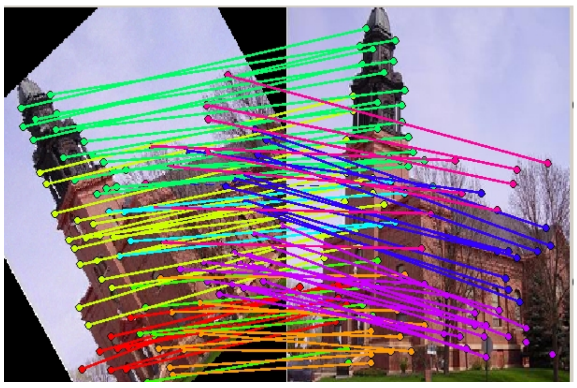

(b) With *SM*

Figure 4: Result of the two-level graph matching approach after 11 iterations


(a) Without *SM*

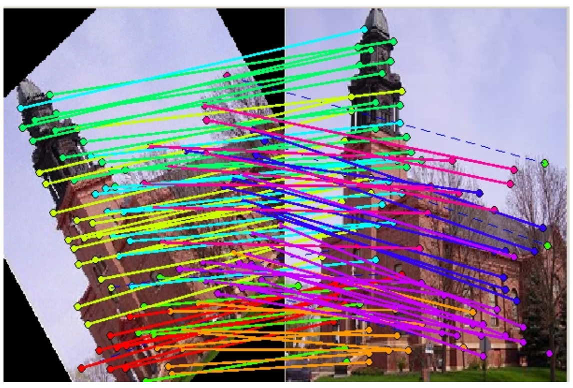
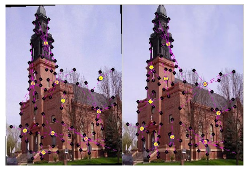
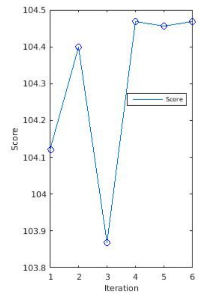
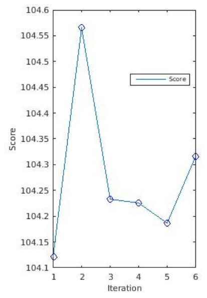
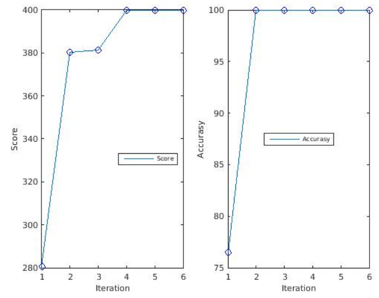
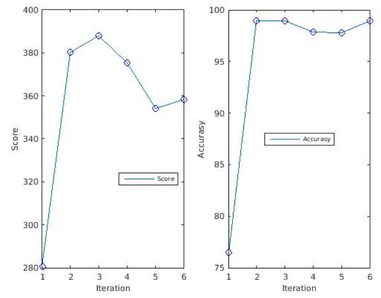
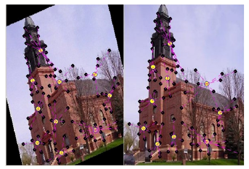
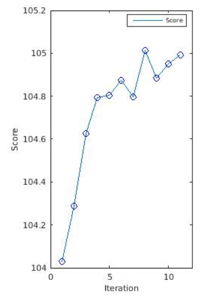
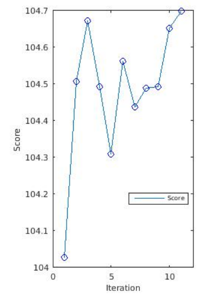
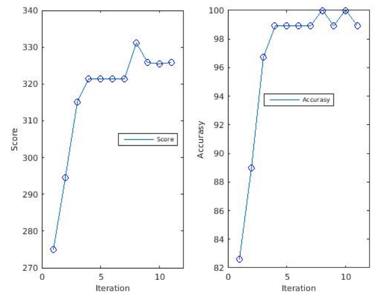
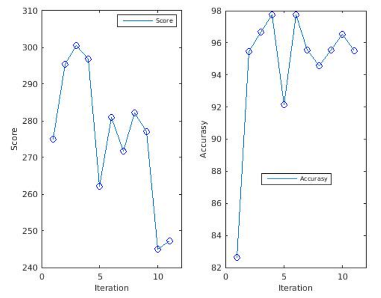
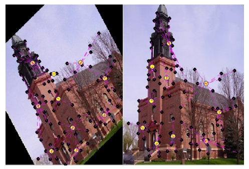
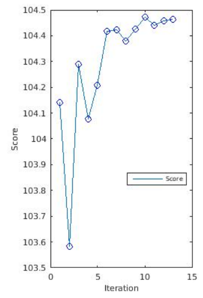
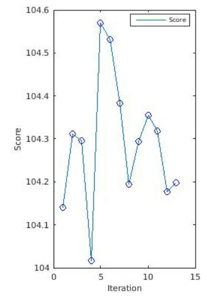
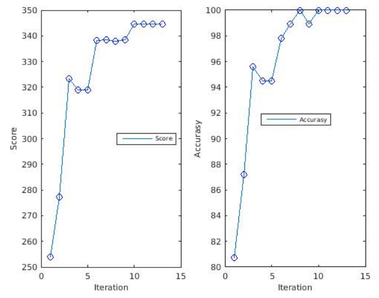
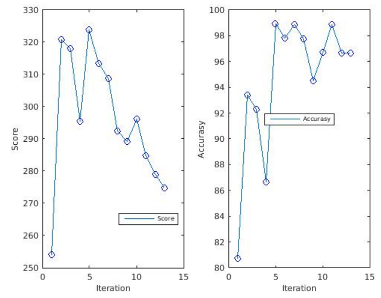

(b) With *SM*

Figure 5: Result of the two-level graph matching approach after 13 iterations

Table 1: Results of two-level Graph Matching Approach

Initial Graps	Higher Level without SM	Higher Leve with SM	Lower Level without SM	Lower Level with SM
				
				
				

4 Issuess

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] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM,*, 24:381–395, 1981.

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

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

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

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

[12] 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.

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

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

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