

A Fast Projected Fixed-Point Algorithm for Large Graph Matching

Yao Lu, Kaizhu Huang, and Cheng-Lin Liu, *Senior Member, IEEE*

Abstract—We propose a fast approximate algorithm for large graph matching. A new projected fixed-point method is defined and a new doubly stochastic projection is adopted to derive the algorithm. Previous graph matching algorithms suffer from high computational complexity and therefore do not have good scalability with respect to graph size. For matching two weighted graphs of n nodes, our algorithm has time complexity only $O(n^3)$ per iteration and space complexity $O(n^2)$. In addition to its scalability, our algorithm is easy to implement, robust, and able to match undirected weighted attributed graphs of different sizes. While the convergence rate of previous iterative graph matching algorithms is unknown, our algorithm is theoretically guaranteed to converge at a linear rate. Extensive experiments on large synthetic and real graphs (more than 1,000 nodes) were conducted to evaluate the performance of various algorithms. Results show that in most cases our proposed algorithm achieves better performance than previous state-of-the-art algorithms in terms of both speed and accuracy in large graph matching. In particular, with high accuracy, our algorithm takes only a few seconds (in a PC) to match two graphs of 1,000 nodes.

Index Terms—Graph matching, projected fixed-point, large graph algorithm, feature correspondence, point matching

1 INTRODUCTION

Graph matching, aiming to find the optimal correspondences between the nodes of two graphs, is an important and active topic in computer vision and pattern recognition [1], [2]. It has been extensively applied in various fields including optical character recognition [3], [4], object recognition [5], [6], shape matching [6], [7], [8], face recognition [9], feature correspondence [10], point matching [11], image retrieval [12], and video indexing [13].

Graph matching is in general an NP-hard discrete optimization problem. Exact graph isomorphism algorithms include Ullman’s method [14], Nauty [15] and VF2 [16], all of which have exponential time complexity in worst cases. To match two graphs within a reasonable time, one has to look for approximate solutions. Moreover, due to noise and variability in real world graphs, the usage of exact graph matching algorithms is very limited. The focus of this paper is the design of a approximate algorithm for efficiently matching general large graphs (e.g. graphs of 1,000 nodes) in computer vision and pattern recognition.

One approach of approximate graph matching algorithms is based tree search [17], [18]. Its basic idea is tree search with backtracking while using heuristics to prune unfruitful paths. Another approach of graph matching algorithms is based on continuous relaxation of the discrete problem while using continuous optimization techniques or heuristics to optimize a matching score. Classic

work include Relaxation Labeling [19], [20], [21], [22] and Graduated Assignment [23]. Generally, continuous relaxation based algorithms have lower computational costs than heuristic search based ones [1].

In this paper, we propose a novel fast graph matching algorithm called Fast Projected Fixed-Point (FastPFP), which is capable of dealing with large graphs of over 1,000 nodes. By using a new projected fixed-point method and a new partial doubly stochastic projection, our algorithm has time complexity $O(n^3)$ per iteration and space complexity $O(n^2)$. In addition to its scalability, our algorithm is easy to implement, robust, and able to match undirected weighted attributed graphs of different sizes. We also proved the linear convergence of the new projected fixed-point algorithm, based on the theory of Convex Projection [37][38]. To the best of our knowledge, our algorithm is the only iterative graph matching algorithm with linear convergence guarantee. We conduct extensive experiments on benchmark datasets of large graphs. FastPFP demonstrated better performance compared to previous state-of-the-art algorithms in most cases (only except for the graph isomorphism case, in which Umeyama’s method is the fastest algorithm). In particular, in a PC, FastPFP is able to match two graphs of 1,000 nodes within a few seconds.

The rest of this paper is organized as follows: In Section 2, we review previous work on continuous relaxation based graph matching algorithms. A simplified analysis of performance of different algorithms is given. In Section 3, we present our formulation of the graph matching problem. In Section 4, we introduce our FastPFP algorithm including the derivation, the new projection method, and convergence analysis. In Section 5, we show extensive experiments conducted on various benchmark datasets in comparison with previous state-

• Yao Lu, Kaizhu Huang and Cheng-Lin Liu are with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
E-mail: yaolubrain@gmail.com, kzhuang@nlpr.ia.ac.cn, liucl@nlpr.ia.ac.cn

of-the-art fast algorithms in large graph matching. In Section 6, we discuss FastPFP’s comparison to FastGA, parameter sensitivity and limitations of our algorithm. Finally, we give concluding remarks in Section 7.

2 PREVIOUS WORK

In this section, we review and analyze continuous relaxation based graph matching algorithms. The main idea of continuous relaxation approach is to relax the original discrete optimization problem into a continuous one and solve it by continuous optimization techniques. For a simplified analysis, let us consider the performance of different algorithms based on continuous relaxation in matching two weighted graphs of n nodes. Relaxation Labeling (RL) [19], [20], [21], [22] is an iterative updating algorithm with the assumption that the likelihood of two nodes to be corresponded can be inferred from their neighboring nodes. The algorithm has time complexity $O(n^4)$ per iteration. Graduated Assignment (GA) [23] is another iterative algorithm based on continuous relaxation. It enjoys a clear optimization objective function and is widely regarded as a state-of-the-art algorithm in the field. Several techniques, including continuous relaxation, linear approximation, softmax and Sinkhorn normalization, are used to optimize this objective function. Compared to RL, GA also has time complexity $O(n^4)$ per iteration but is more accurate and robust [23]. Although the Linear Programming (LP) [24] approach is also based on linear approximation as GA, it has time complexity $O(n^6)$.

The recently proposed new graph matching algorithms include Projections Onto Convex Sets (POCS) [25], Spectral Matching (SM) [26], Spectral Matching with Affine Constraint (SMAC) [27], Integer Projected Fixed Point (IPFP) [29], and Reweighted Random Walks Matching (RRWM) [30]. They all have time complexity $O(n^4)$ (per iteration), since they all require to construct and compute on a compatibility matrix or an association graph, which is an $n^2 \times n^2$ matrix. Probabilistic Graph Matching (PGM) [28] requires the construction of a marginalization matrix, which takes $O(n^4)$ operations as well. Note that, although it is claimed that the compatibility matrix can be very sparse so that sparse matrix techniques can be used for efficient storage and computation, this is not the case for either weighted graphs or densely connected unweighted graphs. Path Following (PATH) algorithm [31] has time complexity $O(n^3)$ per iteration. However, it calls the Hungarian method [32] many times for each iteration. Although the Hungarian method has time complexity $O(n^3)$, the constant factor inside the big-O notation is large. In a PC, it takes more than 500 seconds to apply Hungarian method to a 1000×1000 non-sparse random matrix (See Fig. 2). Therefore, these methods do not have good scalability with respect to graph size. Although some of the above algorithms have high matching accuracy, their high computational

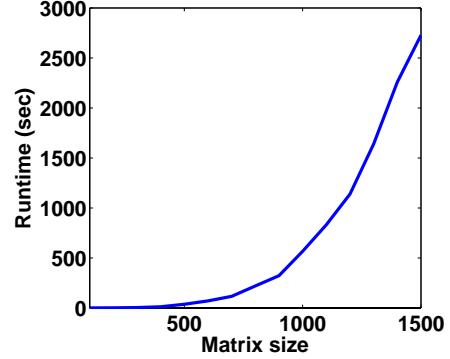


Fig. 1. Runtime of Hungarian method applied to random non-sparse matrices. Matrix size ranges from 100×100 to 1500×1500 .

costs prohibit them from matching large graphs (e.g. graphs of 1,000 nodes) within a reasonable time.

Distinct from the above algorithms, there are three fast graph matching algorithms, all having time complexity $O(n^3)$ (per iteration): Umeyama’s methods [33], FastGA [23] and Projected Gradient [34]. Umeyama’s method uses eigendecomposition and Hungarian methods only once, both of which have time complexity $O(n^3)$. In [31], it was shown that Umeyama’s method is more than an order of magnitude faster than PATH. Despite of its efficiency, Umeyama’s method is unrobust, especially when the two graphs to be matched are far away from isomorphic (see Section 5). Moreover, it can only match graphs of equal sizes, which largely limits its use in practice. GA in general has time complexity $O(n^4)$ per iteration. However, by using a simple trick (see Appendix), GA can be directly reduced to have time complexity $O(n^3)$ per iteration. We call the reduced GA as FastGA. Although IPFP can also be reduced to $O(n^3)$ per iteration using a similar trick, IPFP calls the Hungarian method for each iteration and it takes $5 \sim 10$ iterations for IPFP to converge [29]. Therefore, the reduced IPFP is still not suitable for large graph matching. Projected Gradient (PG) is a classic method to solve constrained optimization problems including graph matching. However, PG can get stuck in poor local optima easily in solving the graph matching problem, as we observed empirically (Section 5). The characteristics of different graph matching algorithms based on continuous relaxation are summarized in Table 1.

The proposed new projected fixed-point algorithm for graph matching is substantially different from the previous projected fixed-point method IPFP in three ways: (1) The projection of IPFP is onto a discrete domain, the space of partial permutation matrices, while the projection of FastPFP is onto a continuous domain, the space of partial doubly stochastic matrices, by using a new projection method [40] (Section 4.3). (2) IPFP requires the expensive Hungarian method for each iteration, making it unsuitable for large graph matching. On the other

TABLE 1
Characteristics of graph matching algorithms based on continuous relaxation

Algorithm	Time Complexity	Space Complexity	Able to Match Graphs of Different Sizes?	Calling Hungarian method for each iteration?	Convergence Guarantee	Convergence Rate
LP	$O(n^6)$	$O(n^4)$	No	No	N/A	N/A
RL	$O(n^4)/\text{iteration}$	$O(n^2)$	Yes	No	Yes	Unknown
GA, POCS	$O(n^4)/\text{iteration}$	$O(n^4)$	Yes	No	Yes	Unknown
RRWM	$O(n^4)/\text{iteration}$	$O(n^4)$	Yes	No	No	N/A
IPFP	$O(n^4)/\text{iteration}$	$O(n^4)$	Yes	Yes	Yes	Unknown
SM, SMAC	$O(n^4)$	$O(n^4)$	Yes	No	N/A	N/A
PGM	$O(n^4)$	$O(n^2)$	Yes	No	N/A	N/A
PATH	$O(n^3)/\text{iteration}$	$O(n^2)$	Yes	Yes	Yes	Unknown
Umeyama	$O(n^3)$	$O(n^2)$	No	No	N/A	N/A
FastGA, PG	$O(n^3)/\text{iteration}$	$O(n^2)$	Yes	No	Yes	Unknown
FastPFP	$O(n^3)/\text{iteration}$	$O(n^2)$	Yes	No	Yes	Linear

hand, each iteration of FastPFP only requires simple arithmetic operations (matrix addition and multiplication). (3) IPFP is proved to converge at finite steps [29] but the convergence rate is unknown. On the other hand, FastPFP is proved to converge at a linear rate.

3 PROBLEM FORMULATION

For two undirected graphs of size n and n' , denote their adjacency matrices (symmetric) by A and A' (binary valued for unweighted graphs and real valued for weighted graphs) and attribute matrices by B and B' , respectively. Each row of B and B' is a k -dimensional vector representing the attributes of a node. The size of A , A' , B and B' are $n \times n$, $n' \times n'$, $n \times k$ and $n' \times k$, respectively. The objective function of graph matching is

$$\min_X \frac{1}{2} \|A - XA'X^T\|_F^2 + \lambda \|B - XB'\|_F^2, \quad (1)$$

$$\text{s.t. } X\mathbf{1} \leq \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \in \{0, 1\}^{n \times n'}, \quad (2)$$

where $\|\cdot\|_F$ is Frobenius matrix norm, λ is a control parameter, and $\mathbf{1}$ is a vector with all its elements equal to one. In (1), the left term can be interpreted as dissimilarity between edges and the right term as dissimilarity between nodes. Constraints in (2) enforces that X is a partial permutation matrix. The constant $\frac{1}{2}$ is for convenience, to be seen later. We assume without losing generality that $n \geq n'$ in the paper. The minimization problem (1)(2) is equivalent to

$$\max_X \frac{1}{2} \text{tr}(X^T AX A') + \lambda \text{tr}(X^T K), \quad (3)$$

$$\text{s.t. } X\mathbf{1} \leq \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \in \{0, 1\}^{n \times n'}, \quad (4)$$

where tr denotes the matrix trace and K denotes the $n \times n'$ matrix BB'^T (See Appendix for derivation). This problem is a Quadratic Assignment Problem (QAP) [36] and is in general NP-hard.

4 ALGORITHM

Our proposed algorithm, FastPFP relies on continuous relaxation of the original discrete problem. The domain of the original discrete problem, the space of partial permutation matrices

$$\{X \mid X\mathbf{1} \leq \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \in \{0, 1\}^{n \times n'}\},$$

is relaxed to onto the space of partial doubly stochastic matrices

$$\{X \mid X\mathbf{1} \leq \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \geq 0\}.$$

A projected fixed-point method is then defined on the relaxed space. In the following subsections, we will give the derivation of our algorithm.

4.1 Fast Projected Fixed-Point

By relaxing the domain of the original QAP problem (3)(4) onto the space of partial doubly stochastic matrices, we have

$$\max_X \frac{1}{2} \text{tr}(X^T AX A') + \lambda \text{tr}(X^T K), \quad (5)$$

$$\text{s.t. } X\mathbf{1} \leq \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \geq 0, \quad (6)$$

where $X \geq 0$ denotes that all elements of X are nonnegative. To derive the algorithm, note that for

$$f(X) = \frac{1}{2} \text{tr}(X^T AX A') + \lambda \text{tr}(X^T K), \quad (7)$$

where $X \in R^{n \times n'}$, the gradient of $f(X)$ is

$$\nabla f(X) = AXA' + \lambda K. \quad (8)$$

In this paper, we introduce a new projected fixed-point algorithm defined as

$$X^{(t+1)} = (1 - \alpha)X^{(t)} + \alpha P_d(\nabla f(X^{(t)})), \quad (9)$$

$$\text{s.t. } \alpha \in [0, 1], \quad (10)$$

where $P_d(\cdot)$ is a partial doubly stochastic projection, defined as

$$P_d(X) = \arg \min_d \|X - d\|_F, \quad (11)$$

$$\text{s.t. } d\mathbf{1} \leq \mathbf{1}, d^T \mathbf{1} = \mathbf{1}, d \geq 0, \quad (12)$$

and α is the step size parameter. For $\alpha = 1$, the algorithm is a straightforward projected fixed-point method. For $0 < \alpha < 1$, the algorithm is a proportionally updated projected fixed-point method. The proportionality introduces more smoothness of the updating process, which could help stabilizing the algorithm (Section 6.2). We call this new algorithm Fast Projected Fixed-Point (FastPFP) due to its linear convergence guarantee (Section 4.4). It also enjoys a property: if the initial state $X^{(0)}$ is a partial doubly stochastic matrix, then each $X^{(t)}$ stays in the space of partial doubly stochastic matrices. This is because for two partial doubly stochastic matrices X and Y , their convex combination

$$Z = (1 - \alpha)X + \alpha Y, \quad \text{s.t. } \alpha \in [0, 1], \quad (13)$$

is another partial doubly stochastic matrix.

4.2 Interpretation and related work

In (9), let $\alpha = 1$ and replace $P_d(\cdot)$ with the projection onto the space of partial permutation matrices, defined as

$$P_{perm}(X) = \arg \min_P \|X - P\|_F^2, \quad (14)$$

$$\text{s.t. } P\mathbf{1} \leq \mathbf{1}, P^T \mathbf{1} = \mathbf{1}, P \in \{0, 1\}^{n \times n'}, \quad (15)$$

which can be solved by the Hungarian method. Then (9) becomes essentially a simplified IPFP

$$X^{(t+1)} = P_{perm}(\nabla f(X^{(t)})). \quad (16)$$

The objective function (3) is increased for each step of IPFP, as proved in [29]. Therefore, FastPFP can be interpreted as a softened version of IPFP.

On the other hand, the Projected Gradient (PG) algorithm is defined as

$$X^{(t+1)} = P_d(X^{(t)} + \alpha \nabla f(X^{(t)})). \quad (17)$$

Although FastPFP and PG are algorithmically similar, in the graph matching problem, PG can get stuck in poor local optima easily. Moreover, PG usually runs significant slower than FastPFP. These experimental findings will be shown in detail in Section 5. And despite of the convergence guarantee of PG [34], the convergence rate of the PG algorithm for solving the graph matching problem is unknown.

In the next subsection, we will discuss how to obtain the solution of the projection $P_d(\cdot)$.

4.3 Partial Doubly Stochastic Projection

The partial doubly stochastic projection can be converted to doubly stochastic projection by introducing slack variables. Let Y be the slacked matrix of X

$$Y = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1n'} & Y_{1(n'+1)} & \dots & Y_{1n} \\ X_{21} & X_{22} & \dots & X_{2n'} & Y_{2(n'+1)} & \dots & Y_{2n} \\ \dots & \dots & \dots & \dots & \dots & & \dots \\ X_{n1} & X_{n2} & \dots & X_{nn'} & Y_{n(n'+1)} & \dots & Y_{nn} \end{pmatrix}. \quad (18)$$

We write the above definition of Y compactly as $Y_{1:n, 1:n'} = X$, where $Y_{1:n, 1:n'}$ denotes the matrix formed by the first n rows and the first n' columns of Y . To project a real nonnegative matrix onto the space of doubly stochastic matrices, the Sinkhorn method [39] is usually used [23], [28], [30], which normalizes each row and column of a matrix alternatively. The objective of the normalization is to find a nearest doubly stochastic matrix D to a matrix Y under the relative entropy measure. Zass and Shashua proposed another doubly stochastic projection [40], which is originally to find a symmetric nearest doubly stochastic matrix D to Y under the Frobenius norm. However their method is also applicable to asymmetric doubly stochastic matrices, if Y is asymmetric. The projection of a matrix Y onto the space of doubly stochastic matrices under the Frobenius norm is the solution to the following problem

$$P_D(Y) = \arg \min_D \|Y - D\|_F, \quad (19)$$

$$\text{s.t. } D\mathbf{1} = \mathbf{1}, D^T \mathbf{1} = \mathbf{1}, D \geq 0. \quad (20)$$

With this doubly stochastic projection, we are able to prove the linear convergence of FastPFP in the next subsection.

The doubly stochastic projection problem can be solved by successive projection. Define two subproblems (projections) of (19-20) as

$$P_1(Y) = \arg \min_D \|Y - D\|_F, \quad \text{s.t. } D\mathbf{1} = \mathbf{1}, D^T \mathbf{1} = \mathbf{1}, \quad (21)$$

$$P_2(Y) = \arg \min_D \|Y - D\|_F, \quad \text{s.t. } D \geq 0. \quad (22)$$

Both P_1 and P_2 have a closed-form solution

$$P_1(Y) = Y + \left(\frac{1}{n} I + \frac{\mathbf{1}^T X \mathbf{1}}{n^2} I - \frac{1}{n} Y \right) \mathbf{1} \mathbf{1}^T - \frac{1}{n} \mathbf{1} \mathbf{1}^T Y, \quad (23)$$

$$P_2(Y) = \frac{Y + |Y|}{2}. \quad (24)$$

The derivation is left in Appendix. The successive projection works as follows: $P_D(Y) = \dots P_2 P_1 P_2 P_1 P_2 P_1(Y)$. The von Neumann successive projection lemma [41] guarantees the successive projection converges to $P_D(Y)$. The solution to the partial doubly stochastic projection under Fronbenius norm is therefore $P_d(X) = P_D(Y)_{1:n, 1:n'}$.

4.4 Convergence Analysis

Now we state the convergence theorem of FastPFP. The proof of the theorem relies on Lemma 1.

Lemma 1: Define $\Omega = \{X | X\mathbf{1} \leq \mathbf{1}, X^T \mathbf{1} = \mathbf{1}, X \geq 0\}$. For two real matrices X_1 and X_2 , $\|P_d(X_1) - P_d(X_2)\|_F \leq \|X_1 - X_2\|_F$.

Proof: Since Ω is a closed convex set, $P_d(\cdot)$ is a nonexpansive projection, which means $\|X_1 - X_2\|_F \geq \|P_d(X_1) - P_d(X_2)\|_F$. See [38] for the proof of nonexpansivity of projections onto convex sets. \square

Theorem 1: Given real matrix $X^{(0)}$, $\alpha \in [0, 1]$ and $\|A \otimes A'\|_2 < \epsilon$, the series

$$X^{(t+1)} = (1 - \alpha)X^{(t)} + \alpha P_d(AX^{(t)}A' + \lambda K), \quad (25)$$

converges at rate $1 - \alpha + \alpha\epsilon$.

Proof: Denote $\text{vec}(X)$, $\text{vec}(Y)$, $\text{vec}(K)$ and $A \otimes A'$ by \mathbf{x} , \mathbf{y} , \mathbf{k} and \mathbf{A} , respectively, where $\text{vec}(\cdot)$ is the vectorization of a matrix and \otimes is the Kronecker product. Then $Y^{(t)} = AX^{(t)}A' + \lambda K$ is equivalent to $\mathbf{y}^{(t)} = \mathbf{Ax}^{(t)} + \lambda \mathbf{k}$. Thus

$$\|X^{(t+1)} - X^{(t)}\|_F \leq (1 - \alpha) \cdot \|X^{(t)} - X^{(t-1)}\|_F \quad (26)$$

$$+ \alpha \cdot \|P_d(AX^{(t)}A' + \lambda K)\|_F \quad (27)$$

$$- P_d(AX^{(t-1)}A' + \lambda K)\|_F. \quad (28)$$

Also

$$\|P_d(AX^{(t)}A' + \lambda K) - P_d(AX^{(t-1)}A' + \lambda K)\|_F \quad (29)$$

$$\leq \|AX^{(t)}A' - AX^{(t-1)}A'\|_F = \|\mathbf{Ax}^{(t)} - \mathbf{Ax}^{(t-1)}\|_2 \quad (30)$$

$$\leq \|\mathbf{A}\|_2 \cdot \|\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}\|_2 \quad (31)$$

$$< \epsilon \cdot \|X^{(t)} - X^{(t-1)}\|_F. \quad (32)$$

Thus

$$\frac{\|X^{(t+1)} - X^{(t)}\|_F}{\|X^{(t)} - X^{(t-1)}\|_F} < 1 - \alpha + \alpha\epsilon. \quad (33)$$

□

Remarks. In practice, we do not have to scale A and A' such that $\|A \otimes A'\|_2 < 1$. We can let $X = X/\max(X)$ at the end of each iteration to prevent numerical instability.

4.5 Discretization

After the convergence of the projected fixed-point method, the resulting matrix X is discretized to obtain the partial permutation matrix P . We use the same discretization method in [26], which is a greedy algorithm, instead of the expensive Hungarian method. This greedy discretization algorithms works as follows:

Step 1. Initialize an $n \times n'$ zero-valued matrix P and a set L containing all its index (i, j) .

Step 2. Given a matrix X , find the index (i^*, j^*) from L such that $X_{i^*j^*} = \arg \max_{(i,j) \in L} X_{ij}$. Set $P_{i^*j^*} = 1$. Remove all indices (i, j) in L that $i = i^*$ or $j = j^*$.

Step 3. Repeat step 2 until L is empty. Return P .

The whole algorithm is summarized in Algorithm 1. Note that $n \geq n'$ is assumed. The matching accuracy of FastPFP is sensitive to initialization. We recommend the initialization $X = \mathbf{1}\mathbf{1}^T_{n \times n'}/nn'$, $Y = \mathbf{0}_{n \times n}$, as we used in all our experiments. Step 9 is needed for numerical stability. For $n = n'$, regardless of fast and sparse matrix computation, Step 3 in Algorithm 1 requires $O(n^3)$ operations per iteration. Step 4 to Step 7 requires $O(n^2)$ operations per iteration. Step 8 requires $O(n^2)$ operations per iteration. Thus, the algorithm has time complexity $O(n^3)$ per iteration and space complexity $O(n^2)$.

Algorithm 1: Fast Projected Fixed-Point

```

Input:  $A, A', K$ 
Output:  $P$ 
1 Initialize  $X$  and  $Y$ 
2 repeat
3    $Y_{1:n, 1:n'} = AXA' + \lambda K$ 
4   repeat
5      $Y = Y + (\frac{1}{n}I + \frac{\mathbf{1}^T Y \mathbf{1}}{n^2} I - \frac{1}{n}Y)\mathbf{1}\mathbf{1}^T - \frac{1}{n}\mathbf{1}\mathbf{1}^T Y$ 
6      $Y = \frac{Y + |Y|}{2}$ 
7   until  $Y$  converges;
8    $X = (1 - \alpha)X + \alpha Y_{1:n, 1:n'}$ 
9    $X = X/\max(X)$ 
10 until  $X$  converges;
11 Discretize  $X$  to obtain  $P$ .

```

5 EXPERIMENTS

In this section, we evaluate the performance in large graph matching of four fast algorithms: FastPFP, Projected Gradient (PG) [17], FastGA and Umeyama's method. All of them have time complexity $O(n^3)$ (per iteration). Some other graph matching algorithms were not compared due to their inability to match large graphs (e.g. graphs of 1000 nodes) in a reasonable time.

For FastGA, FastPFP, and PG, the same greedy discretization described in the above section was used. The convergence criteria are: $\max(|X^{(t+1)} - X^{(t)}|) < \epsilon_1$ or more than I_0 iterations for the first loop (projected fixed-point updating for FastPFP or soft-assignment for FastGA) and $\max(|Y^{(t+1)} - Y^{(t)}|) < \epsilon_2$ or more than I_1 iterations for the second loop (doubly stochastic projection or normalization). For the fairness of comparison, we tuned and fixed the parameters of each algorithm throughout all our experiments. The parameters were well-tuned to achieve the best performance taking both accuracy and efficiency into consideration. For FastPFP, we fixed $\alpha = 0.5$ in all experiments. The sensitivity study of this parameter will be given in the next section. Other parameters were also tried to validate the experimental results. All our experiments were implemented in MATLAB in a 3 GHz Intel Core2 PC.

Remarks. (1) In some datasets, the experiments of Umeyama's method were not conducted due to its inability of matching graphs of different sizes. (2) In the weighted graphs matching experiments, graphs are fully connected. Each node represents a keypoint and each edge represents Euclidean distance between two keypoints. We tried Delaunay triangularization to sparsify the graphs. But the matching quality was poor after applying the sparsification. Therefore, no sparsification was used in the experiments, for simplicity and better matching results.

5.1 Synthetic Random Graphs

In this set of experiments, unweighted random graphs were generated uniformly with 50% connectivity. The

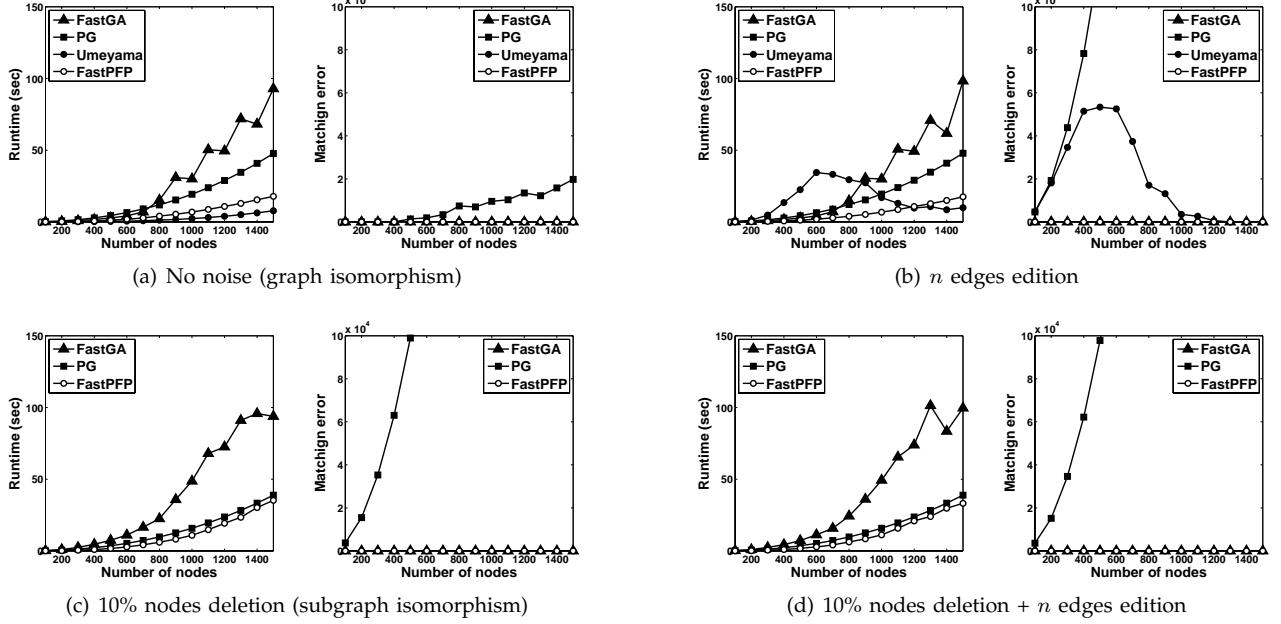


Fig. 2. Synthetic random graph matching

size of the graphs ranges from 100 to 1,500. Four different types of graph matching were tested: two isomorphic graphs with (a) no noise (graph isomorphism), (b) n edges edition (edge edition means if $A_{i,j} = 1$, set $A_{i,j} = 0$ and vice versa), (c) 10% nodes deletion (subgraph isomorphism), (d) n edges edition + 10% nodes deletion. The runtime and matching error (measure by $\|A - X A' X^T\|_F^2 - \|A - X_{GT} A' X_{GT}^T\|_F^2$, where X_{GT} is the ground truth matching matrix) were recorded, as shown in Fig. 2.

As observed from Fig. 2, FastPFP and FastGA were able to find the global optimal solutions in all the above graph matching tasks. But FastPFP is about $3 \sim 5$ times faster than FastGA. PG were stuck in poor local minimum in most cases. Umeyama's method achieved better performance than other algorithms for the isomorphic graph matching in terms of both speed and accuracy. However, its performance was poor in matching non-isomorphic graphs. Roughly speaking, the more isomorphic of two graphs, the better matching results Umeyama's method had. In the left part of Fig. 2(b), the matching error increases with the number of nodes from 100 to 500. This is natural since the numeric range of the matching error increases with the number of nodes given the same percent of connectivity. From 600 nodes on, the matching error decreases with the number of nodes. This is because the graphs are closer to isomorphic as the number of nodes increases since there are only n edges edition for total $O(n^2)$ edges.

5.2 Graphs From Image Sequences

In this set of experiments, weighted graphs were constructed from the CMU House sequence images¹, which has been wildly used for testing graph matching algorithms. In the previous literature, only small graphs (30 nodes) were extracted from the images for matching. To demonstrate scalability of FastPFP, we applied SIFT feature descriptor [42] to all CMU House sequence images. Roughly 800 keypoints were extracted for each image and the number of keypoints varies from image to image. All keypoints in an image were used for the construction of a graph. For each image, a graph whose nodes represent keypoints and edges (matrix A) represent Euclidean distance between keypoints in that image was constructed. For a visualization, we show in Fig. 3 feature correspondence for image 0 vs. image 110. Results of PG, FastGA and FastPFP were shown.

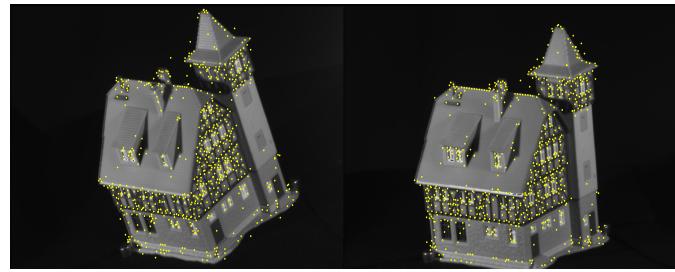
We matched the first image and subsequent images, i.e. image 0 vs. image 1, image 0 vs. image 2, ..., and image 0 vs. image 110. The runtime and matching error (measured as $\|A - X A' X^T\|_F^2$) were recorded for different algorithms. The results are shown in Fig. 4. FastPFP is about 6 times faster than FastGA, with even much lower matching error.

5.3 Graphs From Real Images

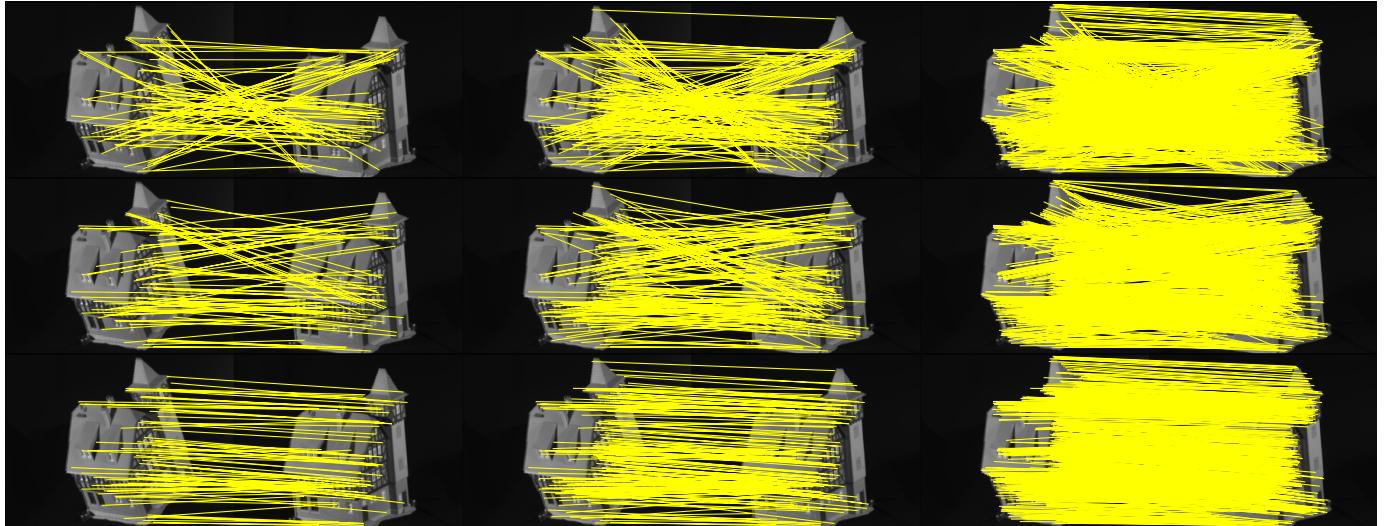
In this set of experiments, attributed and weighted graphs were constructed from real images, chosen from the four image sets: Graffiti, Wall, Bark and Boat². The graphs were constructed as follows. We first applied SIFT descriptor to extract 128-dimensional features for

1. <http://vasc.ri.cmu.edu/idb/html/motion/>

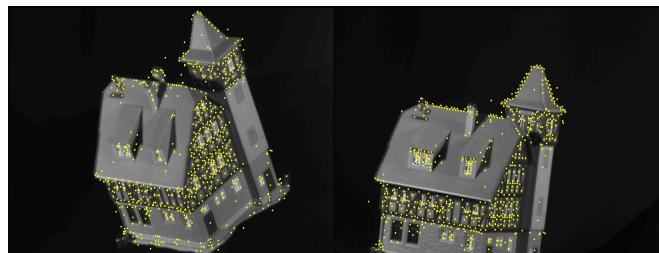
2. <http://www.robots.ox.ac.uk/~vgg/research/affine/>



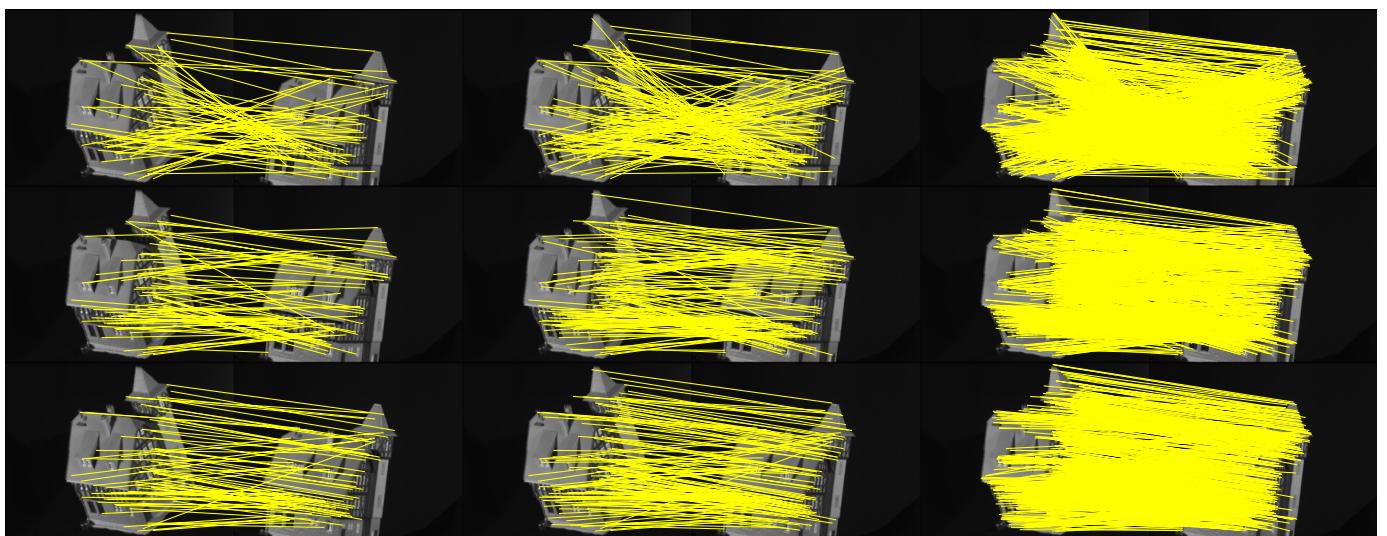
(a) SIFT features (820 nodes vs. 772 nodes)



(b) From top to bottom: PG, FastGA and FastPFP



(c) SIFT features (820 nodes vs. 772 nodes)



(d) From top to bottom: PG, FastGA and FastPFP

Fig. 3. Graphs from CMU house sequence matching. For (c)(f), Left column: 10% matching displayed. Middle column: 20% matching displayed. Right column: 100% matching displayed.

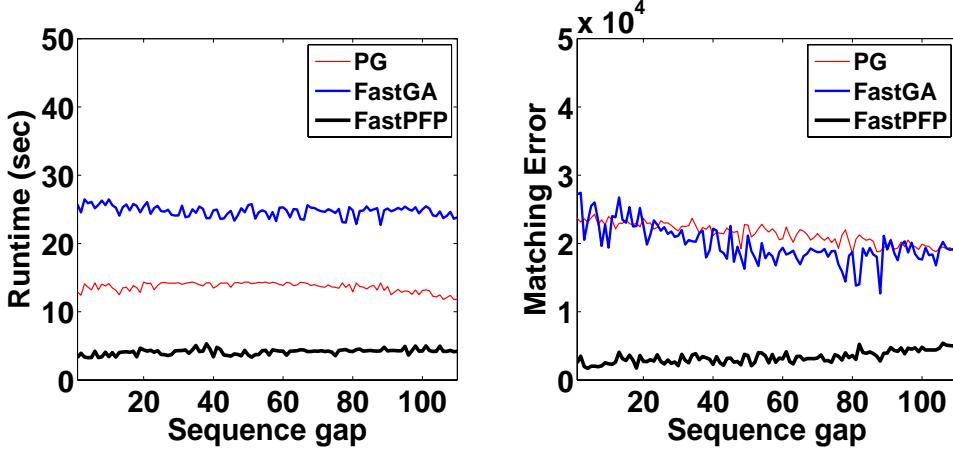


Fig. 4. CMU house sequence

TABLE 2
Graphs from real images: runtime (sec)

Graph pair	PG	FastGA	FastPFP
Graffiti	51.60	87.89	13.52
Wall	19.94	32.25	4.92
Bark	2.84	2.72	0.52
Boat	5.59	9.06	1.36

TABLE 3
Graphs from real images: matching error

Graph pair	PG	FastGA	FastPFP
Graffiti	2.75×10^4	3.98×10^4	5.92×10^3
Wall	1.31×10^4	1.01×10^4	2.39×10^3
Bark	3.07×10^3	2.21×10^3	2.21×10^3
Boat	4.77×10^3	3.18×10^3	1.80×10^3

each image. Then a simple feature selection was performed as follows: a feature of an image was selected if it has the similarity (inner product) to all features of its counterpart image above a threshold. From the selected features, a graph was constructed. Each node represents a keypoint, each node attribute represents a SIFT 128-dimensional feature vector, and each edge represents the Euclidean distance between nodes. Each element K_{ij} of K is set to be the inner product of two SIFT feature vectors i and j . See Fig. 6 for visualization of the matching results for PG, FastGA and FastPFP. The experimental results are listed in Table 2 and 3. The runtime and matching error (measured as $\frac{1}{2}\|A - XA'X^T\|_F^2 + \lambda\|B - XB'\|_F^2$) were recorded. $\lambda = 1$ is used. The sensitivity study of parameter λ is given in Section 6.2. Again, FastPFP demonstrated the best performance: it is about $3 \sim 6$ times faster than FastGA while usually achieving a much lower matching error.

5.4 Graphs From 3D Points

In this set of experiments, weighted graphs were constructed from 3D points. The Face 50 and Stanford

TABLE 4
Graphs from 3D points: runtime (sec)

Graph pair	PG	FastGA	Umeyama	FastPFP
Face pair 1	2.90	15.05	18.40	0.69
Face pair 2	3.06	15.53	18.23	0.69
Stanford Bunny	21.74	118.55	1027.02	8.75

TABLE 5
Graphs from 3D points: matching error (10^5)

Graph pair	PG	FastGA	Umeyama	FastPFP
Face pair 1	6.30	14.3	12.5	0.78
Face pair 2	6.23	16.2	10.6	1.64
Stanford Bunny	29.1	37.4	32.7	9.55

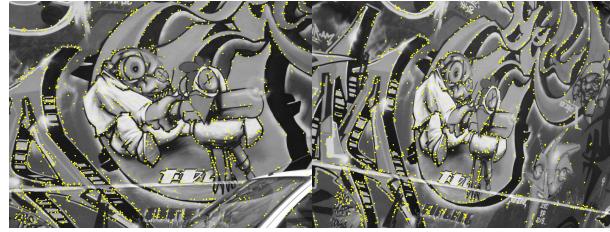
Bunny 3D points datasets³ were used. In the Face 50 dataset, each face has 392 points, all of which were used for the construction of graphs. In the Stanford Bunny 3D data, each set of 3D points was downsampled to 1022 points. Graphs, whose edge represents Euclidean distance between two points, were constructed. The visualization of matching results of PG, FastGA, FastPFP and Umeyama's method are shown in Fig. 7 and 8. The matching error (measured as $\|A - XA'X^T\|_F^2$) and runtime are shown in Table 4 and 5. Again, FastPFP outperformed the others in both speed and accuracy. FastPFP is about $15 \sim 20$ times faster than FastGA.

6 DISCUSSION

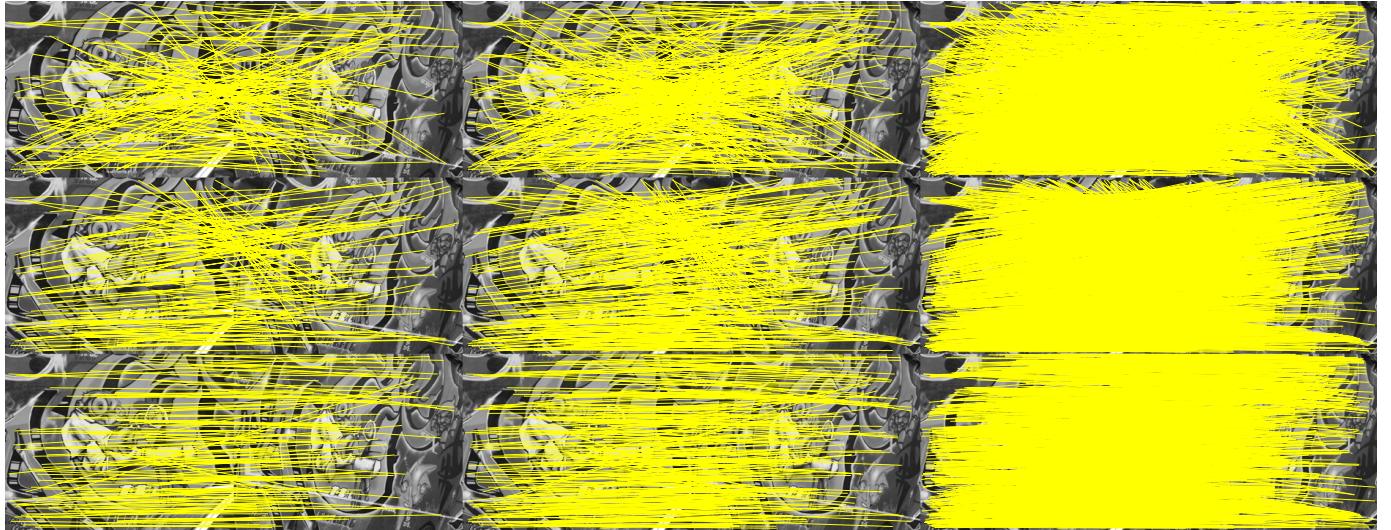
6.1 Comparison To FastGA

Although FastGA and FastPFP both have time complexity $O(n^3)$ per iteration, FastPFP is empirically about $3 \sim 20$ times faster than FastGA. This is mainly due to the slow convergence of FastGA. Despite the theoretical work on convergence of FastGA [35], empirically FastGA does not have good convergence speed, as observed in the experiments. To see that, we show the magnitude

3. Both datasets can be downloaded in <https://sites.google.com/site/myronenko/>



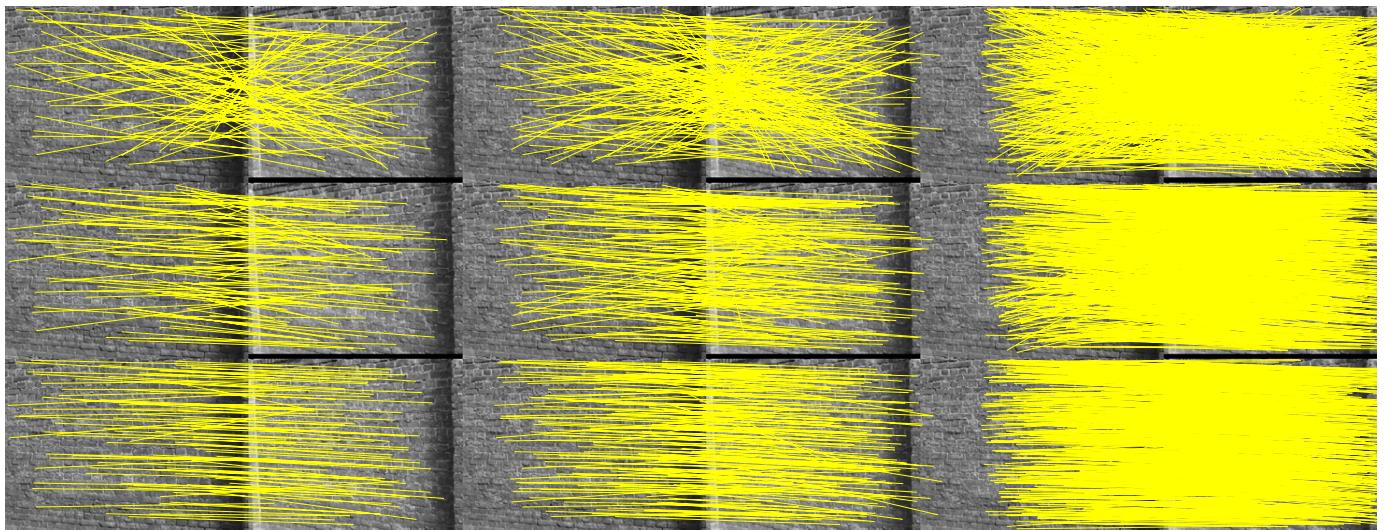
(a) SIFT features (1438 nodes vs. 1443 nodes)



(b) From top to bottom: PG, FastGA and FastPFP

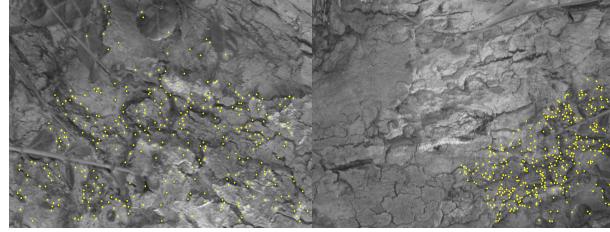


(c) SIFT features (961 nodes vs. 960 nodes)

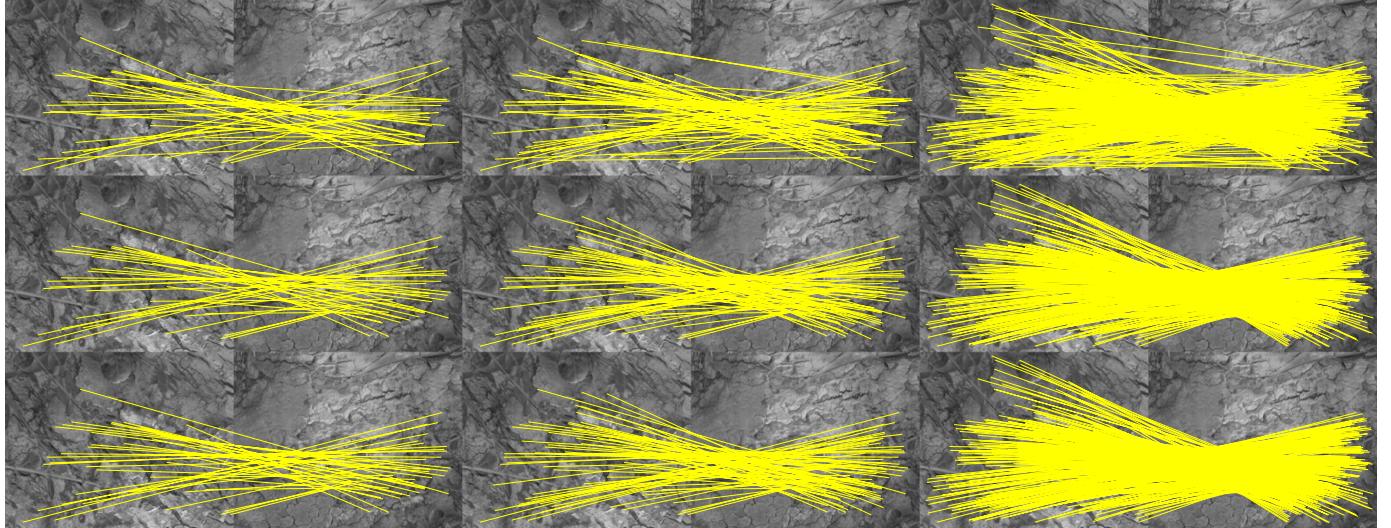


(d) From top to bottom: PG, FastGA and FastPFP

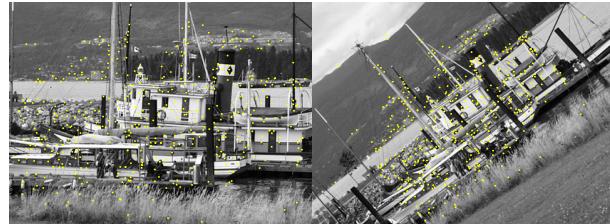
Fig. 5. Graphs from real images matching. For (c)(f), left column: 10% matching displayed. Middle column: 20% matching displayed. Right column: 100% matching displayed.



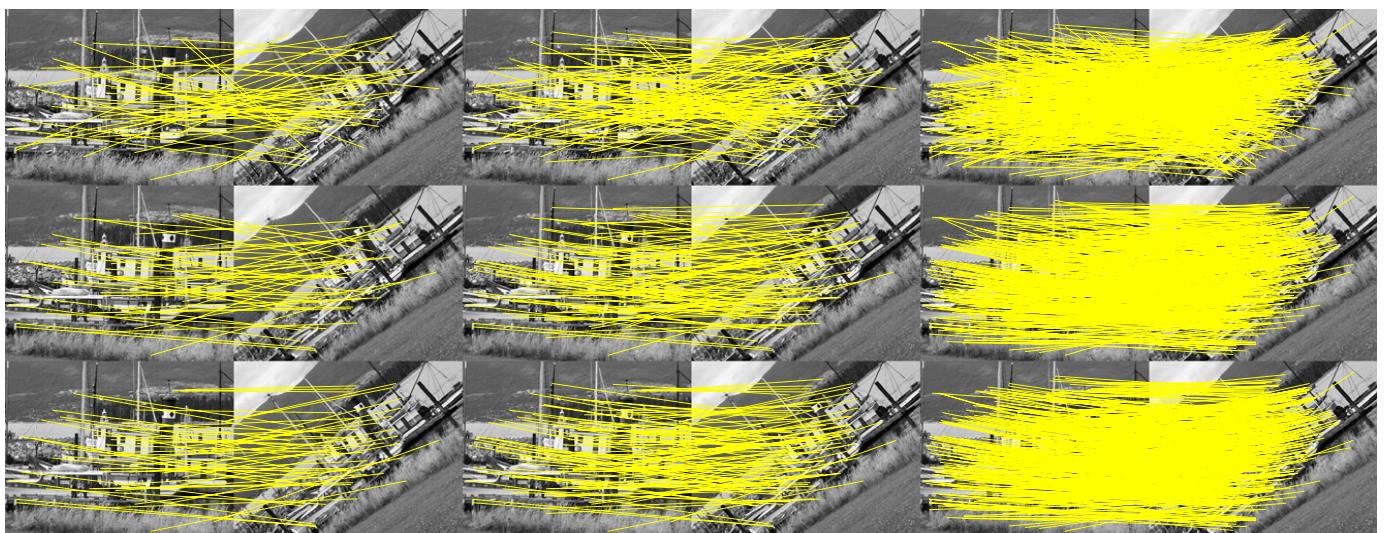
(a) SIFT features (393 nodes vs. 391 nodes)



(b) From top to bottom: PG, FastGA and FastPFP



(c) SIFT features (542 nodes vs. 537 nodes)



(d) From top to bottom: PG, FastGA and FastPFP

Fig. 6. Graphs from real images matching. For (c)(f), left column: 10% matching displayed. Middle column: 20% matching displayed. Right column: 100% matching displayed.

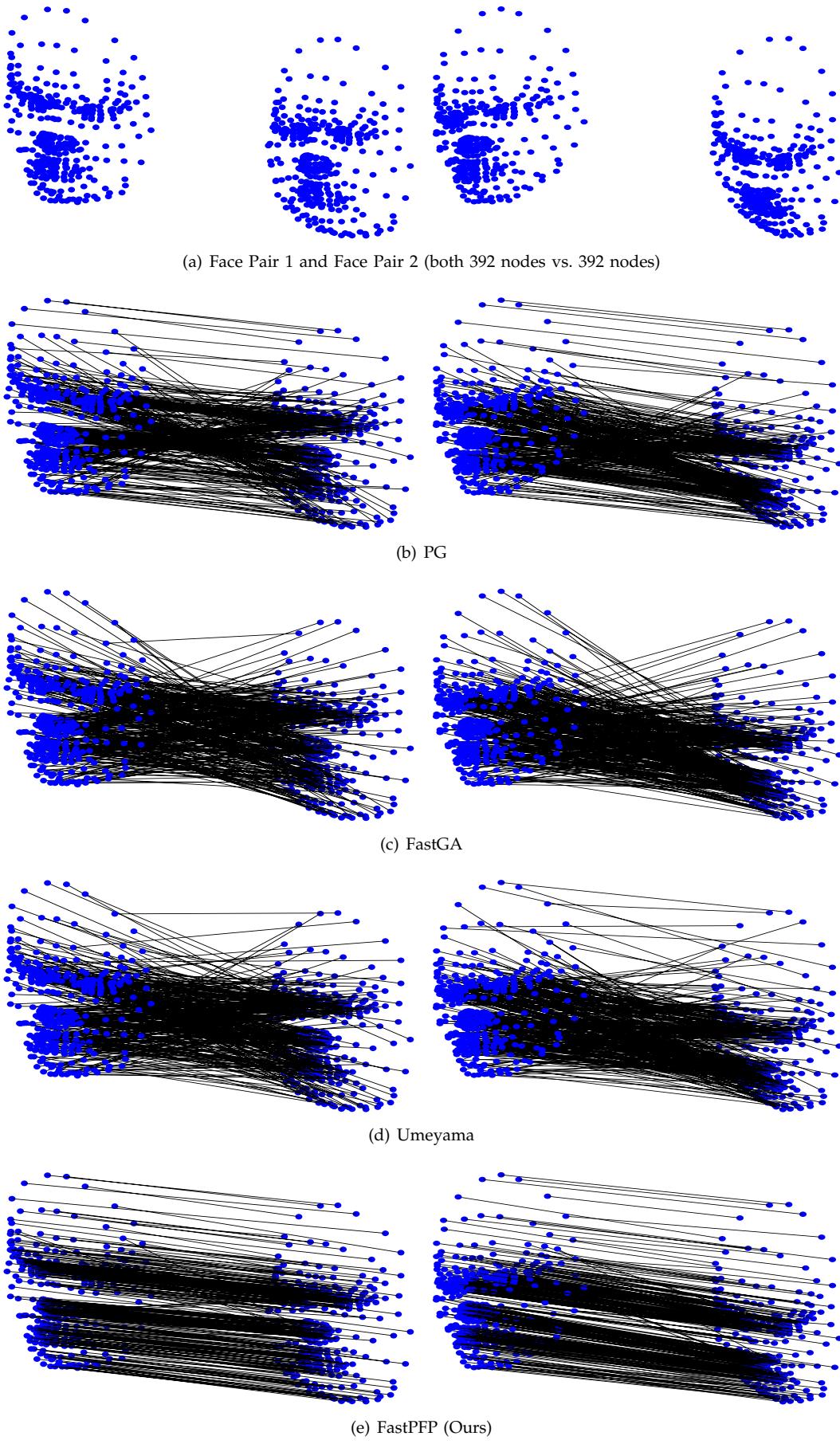


Fig. 7. Graph from 3D points matching

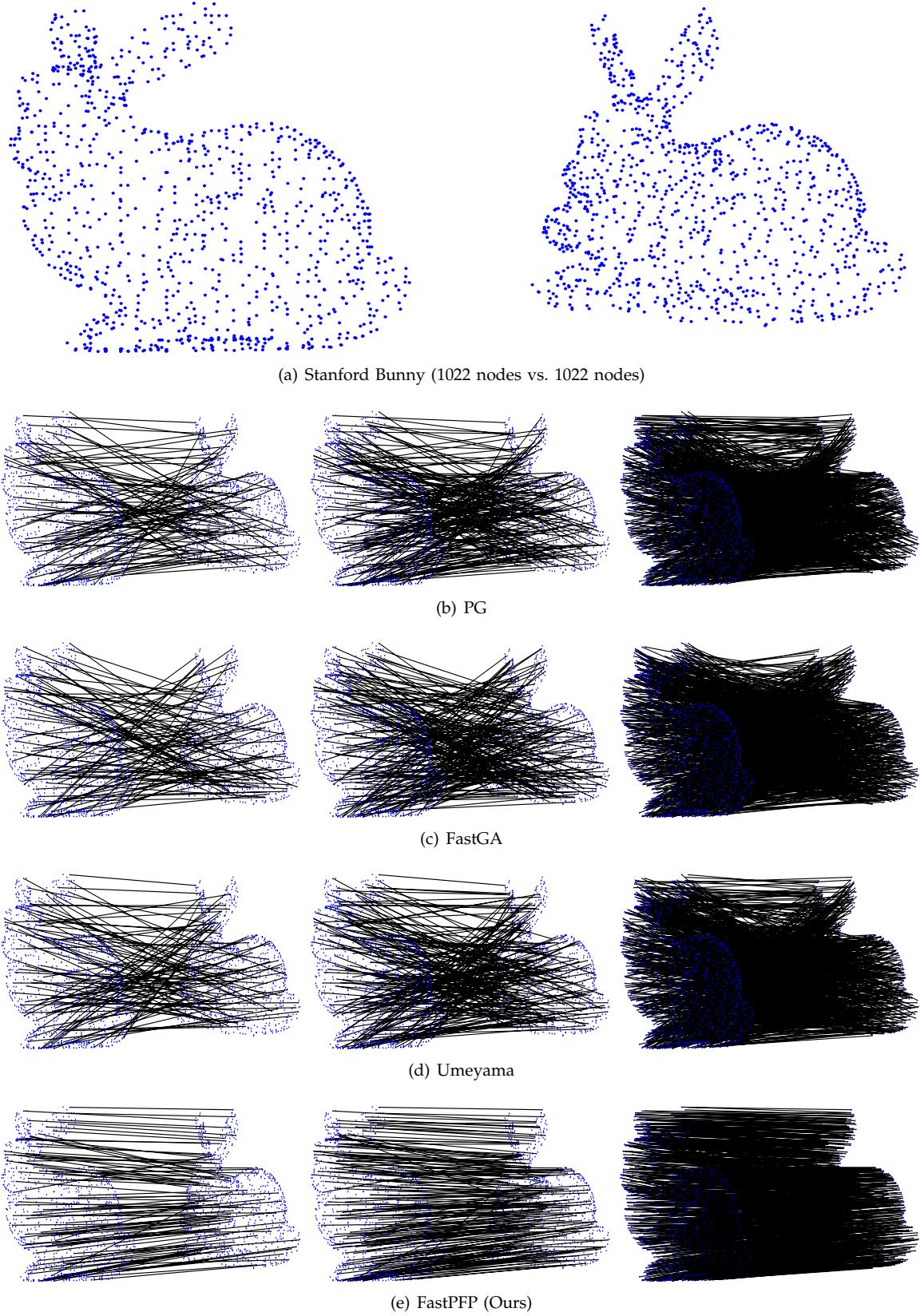


Fig. 8. Graph from 3D points matching: Stanford Bunny. For (b)-(e), left column: 10% matching displayed. Middle column: 20% matching displayed. Right column: 100% matching displayed.

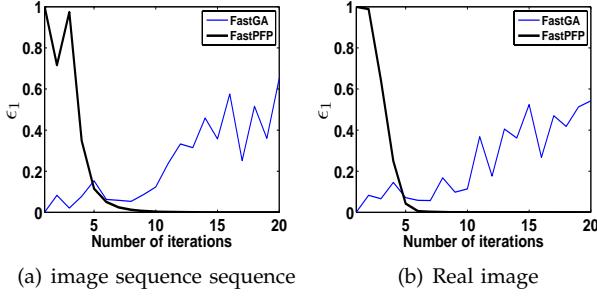


Fig. 9. Convergence of FastPFP and FastGA. The vertical axis represents the value of $\epsilon_1 = \max(|X^{(t+1)} - X^{(t)}|)$.

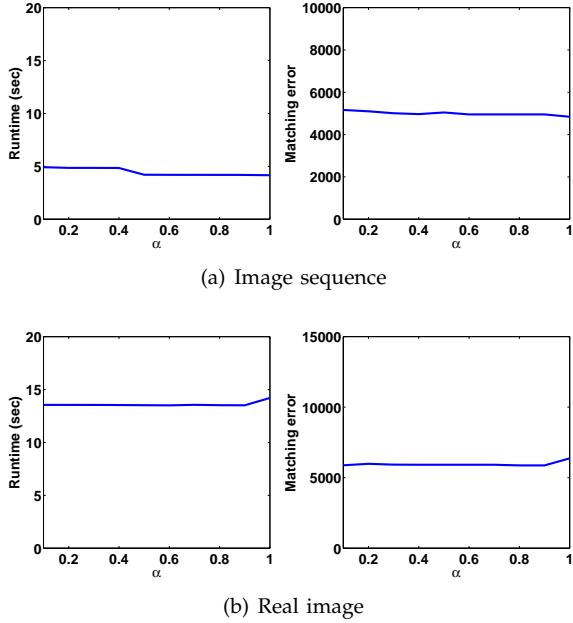


Fig. 10. Performance for different values of α

of parameter $\epsilon_1 = \max(|X^{(t+1)} - X^{(t)}|)$ for image sequence (image 0 vs. image 110) and real image (Graffiti) for 20 iterations in Fig. 9. Besides good convergence, FastPFP only requires simple arithmetic operations (addition, multiplication and maximum element search) while FastGA requires additionally exponentiation and logarithm (for numerically stable implementation).

6.2 Parameter Sensitivity

The performance of FastPFP is not very sensitive to the step size parameter α , for $0 < \alpha < 1$. In Fig. 10, the runtime and matching error of FastPFP in the image sequence (image 0 vs. image 110) and the real image experiment (Graffiti) are shown for different values of α , ranging from 0.1 to 1. The performance of FastPFP is unstable for $\alpha = 1$.

For attributed graph matching, the matching quality is not very sensitive to the parameter λ . We show the matching result for $\lambda = \infty$ (which means the matching objective function becomes $\|B - XB'\|_F^2$ and is solved

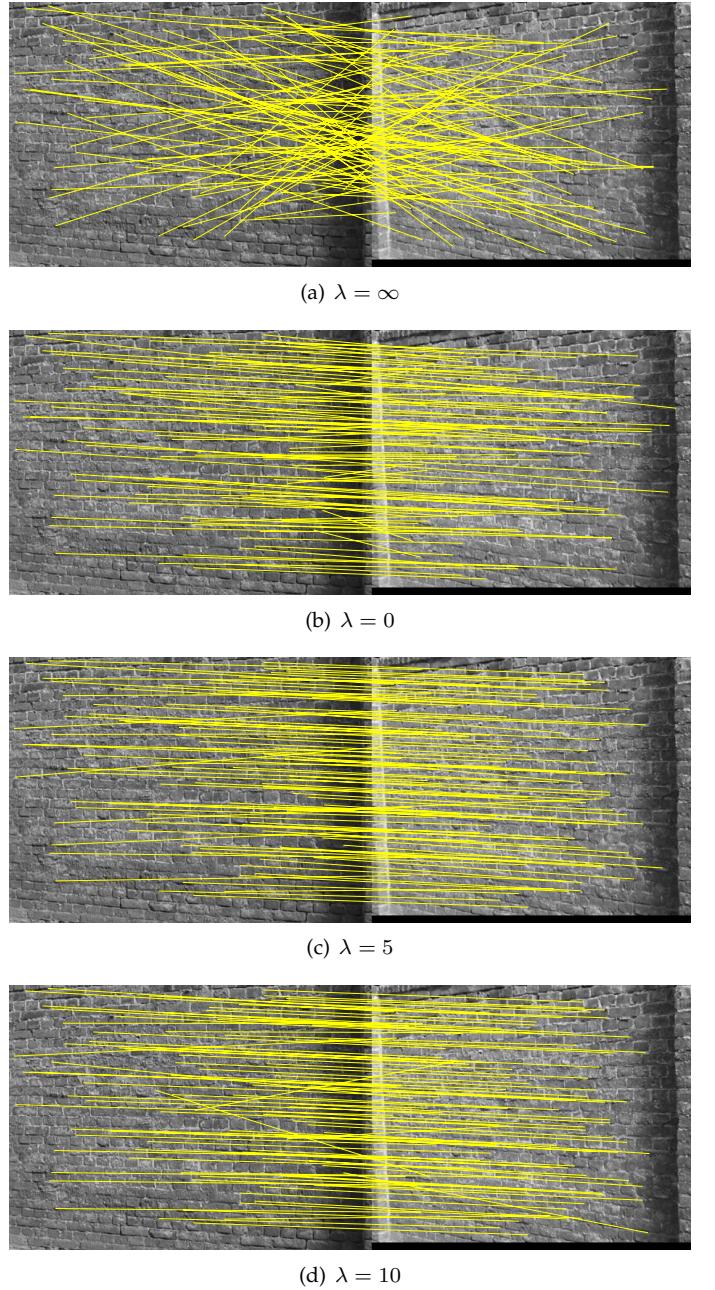


Fig. 11. Performance for different values of λ . 10% of the matching are displayed.

by Hungarian method), $\lambda = 0$ and $\lambda = 10$. The results are similar except for $\lambda = \infty$. The matching results are shown in Fig. 11.

6.3 Limitations

Despite the encouraging experimental results, our algorithm is derived heuristically, as many other approximate graph matching algorithms such as GA (FastGA), Umeyama's method, SM and RWRW. For these algorithms including ours, the optimality of the solutions is not theoretically guaranteed. Also, our algorithm does not use an edge compatibility matrix, due to its high computational cost. Consequently, another limitation of

our algorithm is its inability to handle arbitrary edge relationship (some graphs have vector edge attributes), unlike those algorithms based on edge compatibility matrix.

7 CONCLUSION

We proposed a new fast graph matching algorithm based on a new projected fixed-point method, suitable for large graph matching. Extensive experiments were conducted to demonstrate the strength of our algorithm over previous state-of-the-art algorithms. Future work includes theoretical analysis, extension and applications of our algorithm.

ACKNOWLEDGEMENTS

REFERENCES

- [1] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty Years of Graph Matching in Pattern Recognition," *Int'l J. of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 3, pp. 265-298, 2004.
- [2] H. Bunke and B.T. Messmer, "Recent Advances in Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 169-203, 1997.
- [3] R.S.T. Lee and J.N.K. Liu, "An Oscillatory Elastic Graph Matching Model for Recognition of Offline Handwritten Chinese Characters," *Proc. Third Int'l Conf. Knowledge-Based Intelligent Information Eng. Systems*, pp. 284-287, 1999.
- [4] A. Filatov, A. Gitis, and I. Kil, "Graph-Based Handwritten Digit String Recognition," *Proc. Third Int'l Conf. Document Analysis and Recognition*, pp. 845-848, 1995.
- [5] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509-522, Apr. 2002.
- [6] A.C. Berg, T.L. Berg, and J. Malik, "Shape Matching and Object Recognition Using Low Distortion Correspondences," *IEEE Int'l Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 26-33, 2005.
- [7] K. Siddiqi, A. Shokoufandeh, S.J. Dickinson, and S.W. Zucker, "Shock Graphs and Shape Matching," *Int'l J. Computer Vision*, vol. 35, no. 1, pp. 13-32, 1999.
- [8] B. Huet and E.R. Hancock, "Shape Recognition From Large Image Libraries by Inexact Graph Matching," *Pattern Recognition Letters*, vol. 20, no. 11-13, pp. 1259-1269, Nov. 1999.
- [9] L. Wiskott, J.M. Fellous, N. Krüter, and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775-779, July 1997.
- [10] L. Torresani, V. Kolmogorov, and C. Rother, "Feature Correspondence via Graph Matching: Models and Global Optimization," *Proc. European Conf. on Computer Vision*, pp. 596-609, 2008.
- [11] Y. Zheng and D. Doermann, "Robust Point Matching for Nonrigid Shapes by Preserving Local Neighborhood Structures," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 643-649, April 2006.
- [12] S. Berretti, A.D. Bimbo, and E. Vicario, "Efficient Matching and Indexing of Graph Models in Content-Based Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1089-1105, Oct. 2001.
- [13] K. Shearer, H. Bunke, and S. Venkatesh, "Video indexing and similarity retrieval by largest common subgraph detection using decision trees," *Pattern Recognition*, vol. 34, no. 5, pp. 1075-1091, May 2001.
- [14] J.R. Ullman, "An Algorithm for Subgraph Isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31-42, Jan. 1976.
- [15] B.D. McKay, "Practical Graph Isomorphism," *Congressus Numerantium*, vol. 30, pp. 45-87, 1981.
- [16] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367-1372, Oct. 2004.
- [17] W.H. Tsai and K.S. Fu, "Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Analysis," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 9, no. 12 pp. 757-768, Dec. 1979.
- [18] L.G. Shapiro and R.M. Haralick, "Structural Descriptions and Inexact Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 3, no. 5 pp. 504-519, Sept. 1981.
- [19] M.A. Fischler and R.A. Elschlager, "The Representation and Matching of Pictorial Structures," *IEEE Trans. Computers* vol. 22 no. 1, Jan. 1973.
- [20] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 6, no. 6, pp. 420-433, 1976.
- [21] R.A. Hummel and S.W. Zucker, "On The Foundations of Relaxation Labeling Processes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 3, pp. 267-286, 1983.
- [22] W.J. Christmas, J. Kittler, and M. Petrou, "Structural Matching in Computer Vision Using Probabilistic Relaxation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 749-764, Aug. 1995.
- [23] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377-388, Apr. 1996.
- [24] H. Almohamad and S. Duffuaa, "A Linear Programming Approach for the Weighted Graph Matching Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 5, pp. 522-525, May 1993.
- [25] B.J. van Wyk and M.A. van Wyk, "A POCS-Based Graph Matching Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, Nov. 2004.
- [26] M. Leordeanu and M. Hebert, "A Spectral Technique for Correspondence Problems Using Pairwise Constraints," *Proc. Int'l Conf. Computer Vision*, 2005.
- [27] T. Cour, P. Srinivasan, and J. Shi, "Balanced Graph Matching," *Proc. Advances in Neural Information Processing Systems*, pp. 313-320, 2006.
- [28] R. Zass and A. Shashua, "Probabilistic Graph and Hypergraph Matching," *IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.
- [29] M. Leordeanu, M. Hebert, and R. Sukthankar, "An Integer Projected Fixed Point Method for Graph Matching and MAP Inference," *Proc. Advances in Neural Information Processing Systems*, pp. 1114-1122, 2009.
- [30] M. Cho, J. Lee, and K.M. Lee, "Reweighted Random Walks for Graph Matching," *Proc. European Conf. on Computer Vision*, pp. 492-505, 2010.
- [31] M. Zaslavskiy, F. Bach, and J.P. Vert, "A Path Following Algorithm for the Graph Matching Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2227-2242, Dec. 2009.
- [32] H.W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, nos. 1/2, pp. 83-97, 1955.
- [33] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 695-703, Sept. 1988.
- [34] P.H. Calamai, "Projected Gradient Methods for Linearly Constrained Problems," *Mathematical Programming*, vol. 39, no. 1, Oct. 1987.
- [35] A. Rangarajan, A. Yuille, and E. Mjolsness, "Convergence Properties of the Sofassign Quadratic Assignment Algorithm," *Neural Computation*, vol. 11, pp. 1455-1474, 1999.
- [36] R.E. Burkard, E. Cela, P.M. Pardalos, and L.S. Pitsoulis, "The Quadratic assignment problem," *Handbook of Combinatorial Optimization*, 1998.
- [37] E.H. Zarantonello, "Projections on Convex Sets in Hilbert Space and Spectral Theory," *Contributions to Nonlinear Functional Analysis*, pp. 237-424, Academic Press, New York, 1971.
- [38] D.C. Youla and H. Webb, "Image Restoration by the Method of Convex Projections: Part 1 - Theory," *IEEE Trans. Medical Imaging*, vol. 1, no. 2, Oct. 1982.
- [39] R. Sinkhorn, "A Relationship between Arbitrary Positive Matrices and Doubly Stochastic Matrices," *Annals Math. and Statistics*, vol. 35, pp. 876-879, 1964.
- [40] R. Zass and A. Shashua, "Doubly Stochastic Normalization for Spectral Clustering," *Proc. Advances in Neural Information Processing Systems*, 2007.
- [41] J. Von Neumann, *Functional Operators*, vol. II. Princeton University Press, 1950.

- [42] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. of Computer Vision*, vol. 60, no. 2, Nov. 2004.

APPENDIX

FastGA

In general, GA also has time complexity $O(n^4)$ per iteration. However, when the compatibility matrix $C = A \otimes A'$ (which was mentioned in the original paper of GA [23]) is used, GA has only time complexity $O(n^3)$ per iteration. To see this, consider the most time-consuming step of GA, which has time complexity $O(n^4)$

$$\forall i, \forall a, \quad X_{ia}^{(t+1)} = \exp(\beta Q_{ia}^{(t)}), \quad Q_{ia}^{(t)} = \sum_{b=1}^n \sum_{j=1}^n X_{bj}^{(t)} C_{ajb}, \quad (34)$$

where X is the soft-assignment matrix. If $C_{ajb} = A_{aj}A'_{jb}$, then (34) can be compactly written as

$$X^{(t+1)} = \exp(\beta AX^{(t)}A'), \quad (35)$$

which has time complexity $O(n^3)$.

Problem Formulation

By expanding the Frobenius norm, we have

$$\frac{1}{2} \|A - XA'X^T\|_F^2 + \lambda \|B - XB'\|_F^2 \quad (36)$$

$$= \frac{1}{2} \text{tr}(AA^T) + \frac{1}{2} \text{tr}(X^T X A' X^T X A'^T) - \text{tr}(X^T A X A') \quad (37)$$

$$+ \lambda \text{tr}(BB^T) + \lambda \text{tr}(X^T X B' B'^T) - 2\lambda \text{tr}(X^T BB'), \quad (38)$$

due to the invariance of the matrix trace under cyclic permutation. Since X is a partial permutation matrix and $n \geq n'$, $X^T X = I$. And the first two terms of (37) and (38) are constants. Therefore, the minimization problem

$$\min_X \frac{1}{2} \|A - XA'X^T\|_F^2 + \lambda \|B - XB'\|_F^2, \quad (39)$$

$$\text{s.t. } X\mathbf{1} \leq \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \in \{0, 1\}^{n \times n'}, \quad (40)$$

is equivalent to

$$\max_X \frac{1}{2} \text{tr}(X^T A X A') + \lambda \text{tr}(X^T K), \quad (41)$$

$$\text{s.t. } X\mathbf{1} \leq \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \in \{0, 1\}^{n \times n'}. \quad (42)$$

Successive Projection

The derivation of (23)(24) is due to [40], with slight modification in here. We first consider P_1 . The Lagrangian of (21) is

$$L(D, u_1, u_2) = \text{tr}(D^T D - 2X^T D) \quad (43)$$

$$- u_1^T (D\mathbf{1} - \mathbf{1}) - u_2^T (D^T \mathbf{1} - \mathbf{1}). \quad (44)$$

Let $u_1 = u_2 = u$ and set the derivative with respect to D to zero, we have

$$D = X + u\mathbf{1}^T + \mathbf{1}u^T. \quad (45)$$

Multiplies by $\mathbf{1}$ on both sides: $u = (nI + \mathbf{1}\mathbf{1}^T)^{-1}(I - X)\mathbf{1}$. Combining the fact that $(nI + \mathbf{1}\mathbf{1}^T)^{-1} = (1/n)(I - (1/2n)\mathbf{1}\mathbf{1}^T)$, we obtain (23). The derivation of (24) is straightforward and omitted.



Yao Lu



Kaizhu Huang



Cheng-Lin Liu