Graph Matching with Adaptive and Branching Path Following

Tao Wang, Haibin Ling, Congyan Lang and Songhe Feng

Abstract—Graph matching aims at establishing correspondences between graph elements, and is widely used in many computer vision tasks. Among recently proposed graph matching algorithms, those utilizing the path following strategy have attracted special research attentions due to their exhibition of state-of-the-art performances. However, the paths computed in these algorithms often contain singular points, which could hurt the matching performance if not dealt properly. To deal with this issue, we propose a novel path following strategy, named *branching path following* (BPF), to improve graph matching accuracy. In particular, we first propose a singular point detector by solving a KKT system, and then design a branch switching method to seek for better paths at singular points. Moreover, to reduce the computational burden of the BPF strategy, an *adaptive path estimation* (APE) strategy is integrated into BPF to accelerate the convergence of searching along each path. A new graph matching algorithm named *ABPF-G* is developed by applying APE and BPF to a recently proposed path following algorithm named GNCCP (Liu & Qiao 2014). Experimental results reveal how our approach consistently outperforms state-of-the-art algorithms for graph matching on five public benchmark datasets.

Index Terms—Graph Matching, Path Following, Singular Point, Branch Switching, Adaptive Path Estimation.

1 Introduction

Given two graphs, graph matching involves establishing correspondences between their vertex sets while taking into consideration the consistency between their edge sets as well. It is a fundamental problem in computer science and closely relates to many computer vision problems including feature registration [46], [49], [50], shape matching [5], [41], [53], object recognition [19], [55], visual tracking [8], [52], activity analysis [9], etc. Despite decades of research effort devoted to graph matching, it remains challenging to acquire a global optimum solution in acceptable computational time due to the combinatorial nature over the solutions. A typical way is to utilize relaxation to harness the solution searching, aiming to find approximate optimization methods at the cost of acceptable sub-optimal solutions. Popular approaches include, but not limited to, bipartite matching [43], [47], spectral relaxation [12], [14], [32], continuous optimization [24], [36], [37], [57], [60] and probabilistic modeling [20], [58], to name just a few.

Among recently proposed graph matching algorithms, the ones utilizing the *path following* strategy have attracted much research attention due to their exhibition of state-of-the-art performances [36], [37], [57], [60]. These algorithms reformulate graph matching as a *convex-concave relaxation procedure* (CCRP) problem, which is solved iteratively by interpolating between two relaxed formulations that are simpler than the original one. The solutions of all iterations form a path that recast smoothly the bistochastic matrix solution in the discrete domain. The path following algorithms can be viewed as special cases of the *numerical continuation method* (NCM) [3], which computes approximate solutions of parameterized nonlinear equation systems. These algorithms succeed at *regular points* but may fail at *singular points* (details in Sec. 4). It therefore demands research attention on how to address this issue to improve matching performance.

Motivated by above discussion, we propose a novel path following strategy, named *branching path following* (BPF), to improve path following graph matching algorithms. Roughly speaking, BPF extends the traditional path following strategy by branching new paths at singular points, with three key components. First, it discovers singular points on the original path by analyzing the Jacobian of the associated KKT system. Second, it branches a new path at each singular point using the *pseudo-arclength continuation* method [17], [28]. Third, an *adaptive path estimation* (APE) algorithm is devised that dynamically estimate the iteration steps when exploring each path, and brings further improvement in efficiency. Finally, after searching along these branching paths, BPF chooses the best one in terms of the objective function as the final solution. Note that the original path, which leads to the original path following solution, is always exploited the same as other paths in BPF. As a result, BPF is guaranteed to achieve solutions that are better than, or occasionally as good as, the solutions by the original path following strategy.

Using the proposed strategy, we develop a new graph matching algorithm, named ABPF-G, by applying APE and BPF to the GNCCP (*graduated nonconvexity and concavity procedure*) algorithm [36]. Note that GNCCP is chosen since it is one of the latest path following algorithms, while APE and BPF are by no means limited to working with GNCCP. For a thorough evaluation, we test the proposed ABPF-G algorithm on four popular benchmarks and a synthetic dataset. Experimental results show that, the proposed algorithm significantly improves the path following procedure and outperforms state-of-the-art graph matching algorithms in comparison. Regarding computational cost, ABPF-G achieves similar or even better computational efficiency than GNCCP.

In summary, with the proposed path following strategy for graph matching, this paper makes contribution in four-fold: (1)

- *T. Wang, C.Lang and S. Feng are with Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing 100044, China.*
  E-mail: twang@bjtu.edu.cn, cylang@bjtu.edu.cn, shfeng@bjtu.edu.cn
- *H. Ling is with Computer & Information Sciences Department, Temple University, Philadelphia 19122, USA; and HiScene Information Technologies, Shanghai 201210, China. (Corresponding author: Haibin Ling).*
  E-mail: hbling@temple.edu

we discuss the pitfalls of path following algorithms at singular points, and propose an efficient singular point discovery method; (2) we design a novel branching path following strategy to explore potentially better paths at detected singular points and thus improve matching performance; (3) we devise an adaptive path estimation method for efficiency improvement; and (4) we develop a new graph matching algorithm by applying the proposed strategies to the GNCCP algorithm, and demonstrate the effectiveness of the algorithm in a thorough evaluation. The source code of the proposed algorithm is made available for research usage (http://www.dabi.temple.edu/∼hbling/code/BPF/bpf-2017.htm).

In the rest of this paper, Sec. 2 summarizes related work; then Sec. 3 reviews path following algorithms and Sec. 4 discusses the numerical continuation interpretation; after that, Sec. 5 introduces the proposed APE strategy, and Sec. 6 presents the BPF strategy that employs APE as a component, and further applies them to GNCCP to drive a new algorithm. We finally present experimental validation in Sec. 7 and conclusion in Sec. 8.

2 Related work

Graph matching has been investigated for decades and many algorithms have been proposed. In general, its combinatorial nature makes the global optimum solution hardly available. As a result, approximate solutions are commonly applied to graph matching. In this section we review some sampled classical studies or those closely relate to ours, and leave general graph matching research to two comprehensive surveys [13], [22].

2.1 Classical and recent studies in graph matching

The *graph edit distance* (GED) [45] is one of the most flexible and versatile graph distance metric that arises naturally in the context of inexact graph matching. The idea is to define graph edit operations and use the amount of operations between two graphs to measure their similarity. The exponential complexity, however, restricts the applicability of the edit distance to small graphs. To reduce the complexity, extended algorithms either find approximate the problem with acceptable sub-optimal solutions [27], or find efficient algorithms for restricted classes of graphs such as ordered graph [26]. Instead of acquiring global consistency between graphs, *bipartite graph matching* (BGM) considers only local edge structure during the optimization process [43], and convert the difficult GED problem to a linear assignment problem by encoding the local edge structure into individual nodes.

Another popular way to approximate graph matching is based on spectral relaxation with notable work by Leordeanu and Hebert [32], who propose an efficient spectral solution to correspondence problems using pairwise constraints between candidate assignments. An assignment graph is built such that whose nodes represent the potential correspondences and the weights on the links represent pairwise agreements between potential correspondences. The discovery of the correct correspondences is formulated as a problem of detecting the main strongly connected cluster in the assignments graph, and is then solved using an eigen-analysis approach. Later, the work is extended by Cour et al. [14] in two aspects: (1) to encode the mapping constraints into the spectral decomposition and (2) to apply a bistochastic normalization on the compatibility matrix to considerably reduce matching errors and improve the overall matching performance. Similar to [32], Cho et al. [12] introduce an association graph constructed with nodes as candidate correspondences and edges as pairwise compatibilities between candidate correspondences. The authors cast the search for correspondences between two given graphs as a node ranking and selection problem in the association graph, and introduce an affinity-preserving random walk algorithm to drive the node ranking based on its quasi-stationary distribution. This algorithm is shown to be equivalent to that in [32] for the *integer quadratic programming* (IQP) formulation.

From a different perspective, Zass and Shashua [58] present a probabilistic framework for (hyper-)graph matching. It formalizes a soft matching criterion of the problem input and output, and induces an algebraic relation between the hyperedge weight matrix and the desired node-to-node probabilistic matching. Under this framework, the global optimum of the matching criteria is obtained via an iterative successive projection algorithm. The two lines of spectral relaxation and probabilistic framework somewhat merge in Egozi et al. [20], where the spectral matching scheme of Leordeanu and Hebert [32] is interpreted as a maximum likelihood estimate of the assignment probabilities, given two working assumptions. The authors further propose a new probabilistic formulation of quadratic matching, by relaxing some of the assumptions, to improve the matching performance. These spectral or probabilistic algorithms discussed above drop the discrete constraints in the optimization process, and thus a post discretization step is required to obtain the final solution. Moreover, the employed discretization is generally independent from matching objective optimization, and thus may result in a weakened optimized solution.

Aiming to generate a solution obeying the discrete constraints, some recent studies [2], [25], [44] employ an ℓ_1-norm constraint in the nonnegative domain to induce a sparse (approximate discrete) solution to the matching problem. However, the sparsity of solution acquired by these algorithms is usually uncontrollable, and a post discretization step is also usually required to guarantee the desired discrete solution. Recently, some pure discrete methods [1], [56] are proposed to search the solution directly in the discrete space. These methods can generate a solution obeying the discrete affine mapping constraints, and thus does not require any post-optimization step. However,the optimality of the discrete search method is usually limited which is dependent largely on the initialization [1]. Other sampled recent studies also include [59] that solves the matching between *attributed relational graphs* (ARGs) via mining a hierarchical *and-or graph* (AoG) model, and [40] that analyzes in depth the relaxation in graph matching.

2.2 Path following and related algorithms

Since graph matching is inherently a discrete optimization problem, a typical strategy is to relax it to continuous domain for an initial solution, and then project it into the discrete domain in a graduated and iterative process in which the objective function is employed. Many important algorithms have been designed on top of the relaxation and discretization strategy. For example, Gold and Rangarajan [24] employ the graduated assignment technique to iteratively solve a series of linear approximations of the cost function using Taylor expansion. Leordeanu and Hebert [33] propose an integer projected fixed-point algorithm to optimize the objective function in the integer domain, which takes any continuous or discrete solution, possibly given by some other graph matching method, as input. Subsequently, Lu et al. [39] develop a more efficient projected fixed-point algorithm by utilizing a doubly stochastic projection method.

Along the line there comes the group of *path following* algorithms. In particular, Zaslavskiy et al. [57] reformulate graph matching as a *convex-concave relaxation procedure* (CCRP) problem and then solve it by interpolating between two simpler relaxed formulations, convex relaxation and concave relaxation. More specifically, the *path following* algorithm proposed by them iteratively searches a solution by tracing a path of local minima of a series of functions that linearly interpolate between the two relaxations. Following the similar strategy, Zhou and De la Torre [60], [61] factorize an affinity matrix into a Kronecker product of smaller matrices that encode the structure of the graphs and the affinities between nodes and between edges, and provide specified convex relaxation and concave relaxation based on the factorized matrices. Later, Liu and Qiao [36] propose the *graduated nonconvexity and concavity procedure* (GNCCP), and prove that it equivalently realizes CCRP on partial permutation matrix. GNCCP provides a much simpler way for CCRP without explicitly involving the convex or concave relaxation.

Our work falls into the group of path following algorithms. Compared with previous arts, our method focuses on improving the path following strategy itself, which has not been fully explored before. Preliminary versions of some parts of this work has been published in two conference papers [51], [54], with [54] focusing on the idea of branching path and [51] on adaptive steps. The studies in the two papers are roughly independent, while in this work, we combined their ideas for a further improved graph matching algorithm. As shown in our experiments, such combination benefits not only the computational efficiency but also the matching performance. Details can be found in Sec. 7.

3 Path Following for Graph Matching

3.1 Problem formulation

An undirected graph of n nodes can be represented by $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{v_1, \dots, v_n\}$ and $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ denote the node set and the edge set, respectively. A graph is often conveniently represented by a symmetric adjacency matrix $A \in \mathbb{R}^{n \times n}$, such that $A_{ij} = 1$ if and only if there is an edge between v_i and v_j. The adjacency matrix is usually generalized to a weighted graph by associating a nonnegative real-valued weight $A_{ij} = w_{ij}$ to all pair of nodes. This generalization is important for many applications to capture structural relations among nodes. In the rest of this paper, all adjacency matrices mentioned are weighted with real values unless otherwise stated.

For graph matching, given two graphs $\mathbb{G}^{(i)} = (\mathbb{V}^{(i)}, \mathbb{E}^{(i)})$ of size n_i, $i = 1, 2$, we assume $n_1 \leq n_2$ without loss of generality. The problem is to find a node correspondence $X \in \{0,1\}^{n_1 \times n_2}$ between $\mathbb{G}^{(1)}$ and $\mathbb{G}^{(2)}$ in favor of the following global consistency:

$$\mathcal{E}_1(X) = \sum_{i_1, i_2} c_{i_1 i_2} X_{i_1 i_2} + \sum_{i_1, i_2, j_1, j_2} d_{i_1 j_1 i_2 j_2} X_{i_1 i_2} X_{j_1 j_2}, \quad (1)$$

where $c_{i_1 i_2}$ measures the consistency between the i_1-th node in $\mathbb{G}^{(1)}$ and the i_2-th node in $\mathbb{G}^{(2)}$, and $d_{i_1 j_1 i_2 j_2}$ the the consistency between edge (i_1, j_1) in $\mathbb{G}^{(1)}$ and edge (i_2, j_2) in $\mathbb{G}^{(2)}$. The correspondence matrix X denotes matching result, i.e., $X_{i_1 i_2} = 1$ if and only if $v_{i_1} \in \mathbb{V}^{(1)}$ corresponds to $v_{i_2} \in \mathbb{V}^{(2)}$. In practice, the matching is often restricted to be one-to-one function, which requires $X\mathbf{1}_{n_2} = \mathbf{1}_{n_1}$ and $X^\top \mathbf{1}_{n_1} \preccurlyeq \mathbf{1}_{n_2}$, where $\mathbf{1}_n$ denotes a vector of n ones and $\preccurlyeq$ the element-wise $\leq$.

Let $A^{(i)}$ be the adjacency matrix for $\mathbb{G}^{(i)}$, $i = 1, 2$, a more commonly used formulation for weighted graph matching [4], [57] is defined as

$$\mathcal{E}_2(X) = \mathrm{tr}(C^\top X) + \alpha \| A^{(1)} - X A^{(2)} X^\top \|_F^2, \quad (2)$$

where $C = (c_{i_1 i_2}) \in \mathbb{R}^{n_1 \times n_2}$ is the node dissimilarity matrix, $\alpha \geq 0$ the weight balancing between the node and edge comparisons, and $\| \cdot \|_F$ the Frobenius norm.

Eq. (2) for weighted graph matching is limited in practice because only a scalar attribute is associated to each graph edge and the edge consistency function is restricted to the difference of edge weights. In recent studies [12], [20], [32], [36], [61], a more general formulation of Eq. (1) is formulated in a pairwise compatibility form

$$\mathcal{E}_3(\mathbf{x}) = \mathbf{x}^\top K \mathbf{x}, \quad (3)$$

where $\mathbf{x} \doteq \mathrm{vec}(X) \in \{0,1\}^{n_1 n_2}$ is the vectorized version of matrix X and $K \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ is the corresponding affinity matrix defined as:

$$K_{\mathrm{ind}(i_1,i_2)\mathrm{ind}(j_1,j_2)} = \begin{cases} c_{i_1 i_2} & \text{if } i_1 = j_1 \text{ and } i_2 = j_2, \\ d_{i_1 j_1 i_2 j_2} & \text{else if } A^{(1)}_{i_1 j_1} A^{(2)}_{i_2 j_2} > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

where $\mathrm{ind}(\cdot, \cdot)$ is a bijection mapping a node correspondence to an integer index.

In this paper, we mainly study graph matching algorithms for Eq. (3), since it can encode not only the difference of edge weights but also many complex graph compatibility functions.

3.2 The path following algorithm

In [57], Zaslavskiy et al. introduce the *convex-concave relaxation procedure* (CCRP) into the graph matching problem defined in Eq. (2) by reformulating it as interpolation between two relaxed problems that are simpler than the original one. The first relaxation is obtained by expanding the convex quadratic function $\mathcal{E}_2(X)$ from the set of permutation matrices $\mathcal{P}$ to the set of doubly stochastic matrices $\mathcal{D}$:

$$\min_{X \in \mathcal{D}} \mathcal{E}_2(X) = \mathrm{tr}(C^\top X) + \alpha \| A^{(1)} - X A^{(2)} X^\top \|_F^2. \quad (5)$$

The second relaxation is a concave function

$$\begin{aligned} \min_{X \in \mathcal{D}} \mathcal{E}_4(X) = & -\mathrm{tr}(\Delta X) \\ & - 2\mathrm{vec}(X)^\top \big((L^{(1)})^\top \otimes (L^{(2)})^\top \big) \mathrm{vec}(X), \end{aligned} \quad (6)$$

where Δ is a matrix with element $\Delta_{ij} = \big(D^{(1)}_{ii} - D^{(2)}_{jj}\big)^2$; $D^{(i)}$ and $L^{(i)}$ represent respectively the diagonal degree matrix and the Laplacian matrix of an adjacency matrix $A^{(i)}$, $i = 1, 2$; and $\otimes$ denotes the Kronecker product. A key property of formulation (6) is that, its optimum solution over $\mathcal{D}$ leads to the optimization over $\mathcal{P}$; more importantly, its optimum solution over $\mathcal{P}$ is exactly the solution of the original graph matching problem.

Thus inspired, the *path following* strategy proposed in [57] can be interpreted as an iterative procedure that smoothly projects an initial solution of $\mathcal{E}_2$ in the continuous space $\mathcal{D}$ to the discrete space $\mathcal{P}$ by tracking a path of local minima of a series of functionals $\mathcal{E}_\lambda$ over $\mathcal{D}$

$$\mathcal{E}_\lambda = (1 - \lambda)\mathcal{E}_2 + \lambda \mathcal{E}_4, \quad (7)$$

for $0 \leq \lambda \leq 1$. Each local minimum of $\mathcal{E}_{\lambda + \delta_\lambda}$ is gained by the Frank-Wolfe algorithm [23] given the local minimum of $\mathcal{E}_\lambda$ as the

start point. The convex relaxation guarantees the path starts from the unique local minimum of $\mathcal{E}_2$ for $\lambda = 0$. Increasing λ from 0 to 1, this approach searches toward a local minimum of $\mathcal{E}_4$, and takes it as the final solution. The concave property of $\mathcal{E}_4(X)$ guarantees that the finial solution lies in the discrete space $\mathcal{P}$. More details about the path following algorithm can be found in [57].

4 Numerical Continuation Interpretation

In this section, we interpret the path following algorithms in a numerical continuation view, and then discuss their pitfalls due to singular points. This guides the subsequent improvement on these algorithms. For simplicity, we assume the two graphs to be matched have the same size n, and formulations can be easily extended to handle varied sizes, e.g., by adding dummy nodes.

4.1 KKT system

According to the path following strategy described above and by converting $\mathcal{D}$ into constraints, we need to solve a series of optimization problems with equality and inequality constraints parameterized by λ:

$$\begin{aligned} \mathbf{x}^* &= \arg\max_{\mathbf{x}} \mathcal{E}_\lambda(\mathbf{x}), \\ &\text{s.t.} \left\{ \begin{array}{l} B\mathbf{x} = \mathbf{1}_{2n}, \\ \mathbf{x} \succcurlyeq \mathbf{0}_{n^2}. \end{array} \right. \end{aligned} \tag{8}$$

where $B\mathbf{x} = \mathbf{1}_{2n}$ encodes the one-to-one matching constraints, and $\succcurlyeq$ is the element-wise $\geq$. Specifically, $B \in \{0,1\}^{2n \times n^2}$ can be decomposed into two components $B = [B^{(1)}; B^{(2)}]$, where $B^{(1)}, B^{(2)} \in \{0,1\}^{n \times n^2}$ encode the one-to-one matching constraints on nodes in graphs $\mathbb{G}^{(1)}$ and $\mathbb{G}^{(2)}$ respectively. In particular, $B^{(1)}$ and $B^{(2)}$ are defined as

$$B^{(1)}_{ik} = \left\{ \begin{array}{ll} 1, & \text{if } \exists j \text{ s.t. } k = \mathrm{ind}(i,j), \\ 0, & \text{otherwise.} \end{array} \right. \tag{9}$$

$$B^{(2)}_{ik} = \left\{ \begin{array}{ll} 1, & \text{if } \exists j \text{ s.t. } k = \mathrm{ind}(j,i), \\ 0, & \text{otherwise.} \end{array} \right. \tag{10}$$

where $\mathrm{ind}(\cdot,\cdot)$ is the same bijection mapping as in Eq. (4).

Using Lagrange multipliers α_i, $i = 1, 2, \ldots, 2n$, and KKT multipliers μ_j, $j = 1, 2, \ldots, n^2$, the above constrained problem can be converted to the following unconstrained one

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \Big(\mathcal{E}_\lambda(\mathbf{x}) + \sum_{i=1}^{2n} \alpha_i h_i(\mathbf{x}) - \sum_{j=1}^{n^2} \mu_j g_j(\mathbf{x}) \Big), \tag{11}$$

where

$$h_i(\mathbf{x}) = B(i,:)\mathbf{x} - 1, \text{ and } g_j(\mathbf{x}) = -\mathbf{x}_j. \tag{12}$$

This results in the following system of Karush-Kuhn-Tucker (KKT) equations [30]:

$$\left\{ \begin{array}{rll} \nabla \mathcal{E}_\lambda(\mathbf{x}) + B^\top \alpha + \mu &= \mathbf{0}_{n^2}, & \\ h_i(\mathbf{x}) &= 0, & 1 \leq i \leq 2n, \\ \mu_j g_j(\mathbf{x}) &= 0, & 1 \leq j \leq n^2, \\ \mu_j &\geq 0, & 1 \leq j \leq n^2, \end{array} \right. \tag{13}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{2n})^\top \in \mathbb{R}^{2n}$ and $\mu = (\mu_1, \mu_2, \ldots, \mu_{n^2})^\top \in \mathbb{R}^{n^2}$ are the multiplier vectors.

4.2 Numerical continuation method

The KKT system in Eq. (13) can be reformulated as a constrained nonlinear system

$$F(\lambda, \mathbf{x}, \alpha, \mu) = \mathbf{0}_m, \text{ s.t. } \mu \succcurlyeq \mathbf{0}_{n^2} \tag{14}$$

where $m \doteq 2n^2 + 2n$ is the number of equations in the KKT system. We further introduce vector notation $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)^\top \in \mathbb{R}^m$ such that $\mathbf{u}^\top = [\mathbf{x}^\top, \alpha^\top, \mu^\top]$, then we can have a parameterized nonlinear equations system in the following implicit form:

$$F(\lambda, \mathbf{u}) = \mathbf{0}_m, \tag{15}$$

such that F is a mapping: $F : \mathbb{R}^{m+1} \to \mathbb{R}^m$. The parameter λ is usually a real scalar, and the solution an m-dimensional vector, i.e., $\mathbf{u}$. For a fixed λ, $F(\cdot, \lambda)$ maps m-dimensional Euclidean space into itself. Depending on applications, some or all of the variables may be subjected to inequality constraints, e.g., nonnegative constraints in Eq. (14). The feasible set of inequality constraints is usually defined as $\Omega = \{\mathbf{u} \in \mathbb{R}^m | f(\mathbf{u}) \preccurlyeq \mathbf{0}_m\}$. In path following it can be assumed that the problem is well-defined, i.e., $F(\lambda, \mathbf{u})$ has at least one solution over Ω.

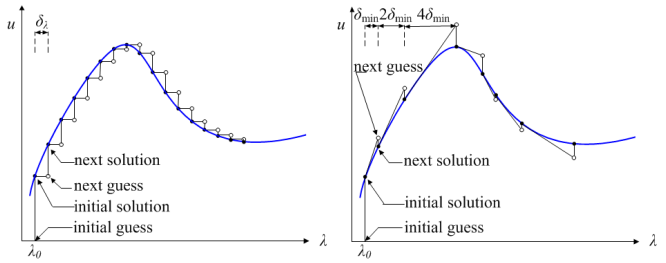
Most solutions of nonlinear equation systems are iterative methods. The existing path following algorithms (PATH [57], FGM [60], GNCCP [36]) can be viewed as special cases of the *numerical continuation method* (NCM) [3]. In general, NCM is a method to compute approximate solutions of a system of parameterized nonlinear equations, by discretizing iteratively trajectories of the flow of $F = \mathbf{0}$. This method works as well in the presence of constraints on some or all of the variables [15], [48].

In fact, the PATH, FGM and GNCCP algorithms correspond to a particular implementation of the so-called *generic predictor corrector* (GPC) approach [3]. GPC is a simple adaptation of the iterative solver to a parameterized problem. As shown in Fig. 1(a), an initial guess λ_0 ($\lambda_0 = 0$ for path following) is acquired to achieve an initial solution $\mathbf{u}_0$, then GPC traces out a solution path by repeatedly incrementing λ until the desired value of λ is reached ($\lambda = 1$ for path following). The solution at a specific λ is used as the initial guess for the solution at $\lambda + \delta_\lambda$, where δ_λ is the step length for each iteration. With δ_λ sufficiently small the iteration applied to the initial guess should converge [3].

4.3 Pitfalls at singular points

An advantage of the GPC approach is that it takes the solution of the original problem as a black box, i.e., all it requires is an initial solution. Simple and intuitive, GPC has been demonstrated to be a reasonable method for solving nonlinear equation systems. This method, however, may fail at points that violate the assumptions of the implicit function theorem [17]. Such a point of potential failure is called a *singular point* when the Jacobian of F (with respect to $\mathbf{u}$) at the point is rank deficient. By contrast, a *regular point* of F is associated with a Jacobian of full rank. Next, we give an intuitive explanation of why and how GPC fails at singular points, and detailed discussion can be found in [17].

A *solution component* $\Gamma(\lambda_0, \mathbf{u}_0)$ of the nonlinear system F is a set of points $(\lambda, \mathbf{u})$ such that $F(\lambda, \mathbf{u}) = \mathbf{0}$ and these points are connected to the initial solution $(\lambda_0, \mathbf{u}_0)$ by paths of solutions. As discussed in [17], near a regular point the solution component Γ is an isolated curve passing through the regular point. By contrast, for a singular point, there may be multiple curves passing through it. The local structure of a point in Γ is determined by high-order derivatives of F. In general, a solution component $\Gamma(\lambda_0, \mathbf{u}_0)$ is a connected component that may contain multiple paths, and these paths intersect at singular points. It is theoretically possible to search all paths in $\Gamma(\lambda_0, \mathbf{u}_0)$ from the given initial point $(\lambda_0, \mathbf{u}_0)$. However, GPC searches along only one path and ignores all



(a) GPC (b) APE

Fig. 1. Illustration of (a) GPC (generic predictor corrector) and (b) the proposed APE (adaptive path estimation). The blue curve indicates the solution path of the parameterized nonlinear equation system.

other ones, and thus may result in a weakly-optimized solution. Therefore, for problems with singular points, more comprehensive handling is desired.

To handle this issue, we propose a *branching path following* (BPF) method to improve the optimality, by first discovering potential singular points and then branching new paths to be searched at discovered singular points, of which the method is detailed described in Sec. 6.

4.4 Computational inefficiency of GPC

In addition to the pitfall at singular points, GPC suffers from high computational burden for two reasons. First, as illustrated in Fig. 1(a), the predictor of GPC is very rough that directly uses the solution at a specific λ as the initial guess for the solution at $\lambda + \delta_\lambda$. The guess is often far away from the actual solution, so the corrector needs to spend much time to search the local optimum at each iteration. Second, the step length δ_λ is fixed and always set to be very small to guarantee reasonable solutions. This naive strategy often leads to unnecessarily many iterations for convergence at desired λ. It therefore demands research attention on how to reduce the computational burden of this approach.

Inspired by the discussion above, we develop a novel *adaptive path estimation* (APE) approach for the parameterized nonlinear equation problem. This approach utilizes a gradient predictor to estimate the solution at each iteration using previous solutions, and dynamically adjust the step length to further accelerate the convergence. We integrate it to the BPF method for graph matching aiming to improve the computational efficiency, and the excellent experimental results validate its effectiveness.

5 Adaptive Path Estimation

In this section we address the efficiency problem in GPC, as pointed out in Sec. 4.4. At each iteration, we develop a gradient predictor to estimate the solution using previous solutions, instead of using directly last solution, and then dynamically adjust the step length to further accelerate the convergence. Since we are only interested in solution $\mathbf{x}$ and it is independent of Lagrange multipliers α_i and KKT multipliers μ_i in path estimation and step length adjustment, we quote $\mathbf{x}$, rather than $\mathbf{u}$, for description simplicity.

5.1 Path prediction

From the implicit function theorem [29], the solutions of the parameterized nonlinear equation system $F(\lambda, \mathbf{x}, \alpha, \mu) = \mathbf{0}$ form a smooth path parameterized by λ. We thus are able to predict the t-th solution using previous solutions. In particular, we propose to predict current solution using the differences of the previous solutions.

Denote $\mathbf{x}_i$ the solution at a previous iteration i $(i < t)$, the solution difference is computed as $\mathbf{d}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$. We predict the difference $\mathbf{d}_t$ and the solution $\mathbf{x}'_t$ at iteration t as

$$\mathbf{d}_t = \frac{\sum_{i=t-k}^{t-1} w_i \mathbf{d}_i}{\sum_{i=t-k}^{t-1} w_i}, \tag{16}$$

$$\mathbf{x}'_t = \mathbf{x}_{t-1} + \mathbf{d}_t, \tag{17}$$

where k controls the size of the window for approximation, and w_i the weight of the differences at previous solutions. We prefer Gaussian weights as $w_i = \exp(-(t-i)^2/\gamma)$ and set $\gamma = 10k$ throughout our experiments. The motivation behind this prediction is the smoothness of the path, which implies the similarity between solution differences of consecutive iterations.

Once the prediction $\mathbf{x}'_t$ is ready, it is used as the start point to search for the local optimum $\mathbf{x}_t$ via the Frank-Wolfe algorithm. Compared with $\mathbf{x}_{t-1}$, which is used in the original GPC approach as the start point, $\mathbf{x}'_t$ is in general closer to $\mathbf{x}_t$ and thus the search procedure is more efficient.

5.2 Adaptive step length

It is usually hard to choose a proper step length δ_λ of each iteration. Too small steps lead to high computational burden while too large ones may hurt matching accuracy. To address the issue, we propose using adaptive step lengths. Denote δ_{λ_t} the step length for iteration t, and δ_{λ_t} should depend on the estimation $\mathbf{x}'_t$. When $\mathbf{x}'_t$ is very close to the local optimum $\mathbf{x}_t$, suggesting high path smoothness around $\mathbf{x}_t$, we can afford using a large δ_{λ_t}; by contrast, when $\mathbf{x}'_t$ is far from $\mathbf{x}_t$, a small δ_{λ_t} is more reasonable.

We use an *adaption rate* ρ to dynamically adjust the step length. Initially, we set the step length as $\delta_{\lambda_0} = \delta_{\min}$, where $\delta_{\min}$ represents the minimum step length allowed. Then, at each iteration t, δ_{λ_t} increases or decreases by ρ times depending on the similarity between $\mathbf{x}'_t$ and $\mathbf{x}_t$. Specifically, we have

$$\delta_{\lambda_t} \leftarrow \begin{cases} \rho\delta_{\lambda_{t-1}}, & \text{if } \mathrm{tr}((\mathbf{x}_t - \mathbf{x}'_t)^\top(\mathbf{x}_t - \mathbf{x}'_t)) \leq \theta \\ \max(\delta_{\min}, \frac{\delta_{\lambda_{t-1}}}{\rho}), & \text{otherwise} \end{cases} \tag{18}$$

where θ is the threshold of the error tolerance of the estimation. Once the update on δ_{λ_t} is done, we run the optimization (Frank-Wolfe) for the new value $\lambda + \delta_{\lambda_t}$. The idea behind this adaptation schema is to choose δ_{λ_t} which keeps the estimation $\mathbf{x}'_t$ close enough to the real local optimum $\mathbf{x}_t$.

With the adaptive step length, the difference predictor in Eq. (17) is extended to

$$\mathbf{d}_t = \frac{\sum_{i=t-k}^{t-1} (w_i \mathbf{d}_i / \delta_{\lambda_{i-1}})}{\sum_{i=t-k}^{t-1} w_i}, \tag{19}$$

$$\mathbf{x}'_t = \mathbf{x}_{t-1} + \mathbf{d}_t \delta_{\lambda_{t-1}}. \tag{20}$$

The adaptive path estimation method is illustrated in Fig. 1(b) and summarized in Alg. 1, which takes path prediction (line 5) and adaptive step length (lines 13 to 17) as components. With accurate predictor and adaptive step length, APE gains much higher computational efficiency than the original GPC approach. Interestingly, APE by itself also boosts matching performance, which is discussed in Sec. 6.4 and illustrated in experiments.

Algorithm 1 Adaptive Path Estimation $(\lambda_0, \mathbf{x}_0)$

% $(\lambda_0, \mathbf{x}_0)$: initial point of the path.
% $\mathcal{X} = \{\mathbf{x}_0, ..., \mathbf{x}_{t-1}\}$: solutions of previous iterations.
% $\Lambda = \{\delta_{\lambda_0}, ..., \delta_{\lambda_{t-1}}\}$: step lengths.
% $\delta_{\min}$: parameter of minimum step length.
1: Initialization: $\delta_{\lambda_0} \leftarrow \delta_{\min}, \lambda \leftarrow \lambda_0, t \leftarrow 0$
2: Initialization: $\mathcal{X} \leftarrow \{\mathbf{x}_0\}, \Lambda \leftarrow \{\delta_{\lambda_0}\}$
3: **while** $\mathbf{x}_t$ not discretized **do**
4: $t \leftarrow t + 1$
5: $\mathbf{x}'_t \leftarrow \mathbf{x}_{t-1} + \delta_{\lambda_{t-1}} \frac{(\sum_{i=t-k}^{t-1} (w_i \mathbf{d}_i / \delta_{\lambda_{i-1}}))}{(\sum_{i=t-k}^{t-1} w_i)}$ % Prediction
6: $p \leftarrow \mathbf{x}'_t$ % start point to search local optimum
7: **while** p not converged **do**
8: $y = \arg\min_y \mathrm{tr}(\nabla \mathcal{E}_\lambda(p)^T y)$, s.t. $y \in \mathcal{D}$
9: $\beta = \arg\min_\beta \mathcal{E}_\lambda(p + \beta(y - p))$, s.t. $0 \le \beta \le 1$
10: $p \leftarrow p + \beta(y - p)$
11: **end while** % Frank-Wolfe algorithm
12: $\mathbf{x}_t \leftarrow p$, $\mathbf{d}_t \leftarrow \mathbf{x}_t - \mathbf{x}_{t-1}$
13: **if** $\mathrm{tr}((\mathbf{x}_t - \mathbf{x}'_t)^\top (\mathbf{x}_t - \mathbf{x}'_t)) \le \theta$ **then**
14: $\delta_{\lambda_t} \leftarrow \rho \delta_{\lambda_{t-1}}$
15: **else**
16: $\delta_{\lambda_t} \leftarrow \max(\delta_{\min}, \frac{\delta_{\lambda_{t-1}}}{\rho})$
17: **end if** % Adaptive step length
18: $\lambda \leftarrow \lambda + \delta_{\lambda_t}$
19: $\mathcal{X} \leftarrow \mathcal{X} \cup \{\mathbf{x}_t\}$
20: $\Lambda \leftarrow \Lambda \cup \{\delta_{\lambda_t}\}$
21: **end while**
22: **return** $\mathcal{X}$

Algorithm 2 Branching Path Following.

% T: the set of candidate solutions, i.e., end points of branching paths.
% $P_i = \{\mathbf{u}_{i0}, \ldots, \mathbf{u}_{iN_i}\}$: the i-th branching path with $N_i + 1$ solution points.
1: Initialize a start point $(\lambda_0, \mathbf{u}_0)$
2: Initial path $P_0 \leftarrow \mathrm{APE}(\lambda_0, \mathbf{u}_0)$ % Adpative path estimation
3: $T \leftarrow \{\mathbf{u}_{0N_0}\}$
4: Discover the set of singular points S from P_0
5: $i \leftarrow 0$
6: **for** each point $(\lambda, \mathbf{u})$ in S **do**
7: $i \leftarrow i + 1$
8: Compute the solution $(\lambda^*, \mathbf{u}^*)$ of Eq. (23)
9: $P_i \leftarrow \mathrm{APE}(\lambda^*, \mathbf{u}^*)$ % Adpative path estimation
10: $T \leftarrow T \cup \{\mathbf{u}_{iN_i}\}$
11: **end for**
12: Select the best $\mathbf{u}^*$ from T in line with the objectives
13: Return $\mathbf{u}^*$ as the final solution

6 BRANCHING PATH FOLLOWING

Although APE gains much higher computational efficiency than the original GPC approach as discussed above, it still meets the problem at singular points as GPC. In this section, we propose the *branching path following* (BPF) strategy that branches new paths at singular points toward potentially better solutions for nonlinear equation systems. The full BPF algorithm is summarized in Alg. 2.

Based on the numerical continuation method, BPF is a general framework for parameterized nonlinear equation systems. Specially, we apply BPF to the graph matching problem in this paper. As described in Alg. 2, BPF contains three main components:

- **Singular point discovery (line 4):** Given an initial solution path, it finds potential singular points by determining the signs of determinants of Jacobian on consecutive sampled points.
- **Branch switching (line 8):** It branches new paths to be searched at discovered singular points by re-parameterizing the problem with an introduced arclength parameter.
- **Adaptive path estimation (lines 2 and 9):** Starting from a given initial point, it searches along the path with adaptive steps based on path estimation.

We first discuss each component of BPF in detail in the following subsections, and subsequently apply BPF to GNCCP to develop a new graph matching algorithm, followed by analysis of optimality and computational complexity.

6.1 Singular points discovery

It is an essential step for BPF to discover singular points. Theoretically, these points can be detected by checking whether the Jacobian of F is of full rank. However, it is impractical to check discrete samples by sampling λ_i, since these samples are rarely able to cover the exact singular points.

Denote the Jacobian of F at specific solution point $(\lambda, \mathbf{u})$ as

$$J_\lambda = F_{\mathbf{u}}(\lambda, \mathbf{u}) \in \mathbb{R}^{m \times m}. \tag{21}$$

A singular point $(\lambda, \mathbf{u})$ should have determinant $|J_\lambda| = 0$. A reasonable assumption is that the curve formed by $(\lambda, |J_\lambda|)$ over λ is continuous. This implies there is at least one singular point between two points $(\lambda_1, \mathbf{u}_1)$ and $(\lambda_2, \mathbf{u}_2)$ if $|J_{\lambda_1}|$ and $|J_{\lambda_2}|$ have different signs.

Thus inspired, we design a simple yet effective way for singular point discovery by checking the signs of determinants of Jacobian on consecutively sampled points. Specifically, denote $(\lambda_t, \mathbf{u}_t)$ the point at iteration t in the path, we have at least one singular points between $(\lambda_t, \mathbf{u}_t)$ and $(\lambda_{t+1}, \mathbf{u}_{t+1})$ if $|J_{\lambda_t}||J_{\lambda_{t+1}}| \le 0$. Since our goal is to branch a new path, it is not necessary for us to locate the exact position of the singular point. We mark $(\lambda_t, \mathbf{u}_t)$ as an approximate singular point, and use is to compute a new point $(\lambda^*_{t+1}, \mathbf{u}^*_{t+1})$ in the new path, which is taken as the initial solution for searching along the new path. The method of finding the new path is described in the next subsection.

The determinant of a Jacobian matrix J_λ is computed through LU decomposition $J_\lambda = LU$, where L is a lower unitriangular matrix and U an upper triangular matrix. Since we are only interested in the sign of $|J_\lambda|$, it can be decided by accumulating the signs of the diagonal elements of U as $\prod_i \mathrm{sign}(U_{ii})$, where $\mathrm{sign}(\cdot)$ is the sign function defined as

$$\mathrm{sign}(a) = \begin{cases} 1, & \text{if } a > 0, \\ -1, & \text{if } a < 0, \\ 0, & \text{if } a = 0. \end{cases} \tag{22}$$

6.2 Branch switching

Finding the solution path passing a singular point is called branch switching. Our goal is to branch new paths at singular points and then search the best solution along these branched paths.

Once a singular point $(\lambda_t, \mathbf{u}_t)$ is discovered, we branch a new path using the *pseudo-arclength continuation* (PAC) algorithm [17], [28]. PAC is inspired by the observation that an ideal parameterization of a curve is arclength-based. It provides a

standard strategy to remedy the failure of GPC (and the proposed APE) at simple folds, which happened typically at singular points (for details about PAC and simple fold, please see [17]). More specifically, PAC reparameterizes path following by introducing the arclength parameter l, so that both $\mathbf{u}$ and λ depend on l. With the parameterization, we extend equations (13) to the following

$$G(\lambda, \mathbf{u}, l) = \left\{ \begin{array}{c} F(\lambda, \mathbf{u}) \\ N(\lambda, \mathbf{u}, l) \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{0}_m \\ 0 \end{array} \right\}, \tag{23}$$

where the normalization equation $N(.) = 0$ approximates the statement that l represents arclength. Denote $(\dot{\lambda}, \dot{\mathbf{u}})$ the tangent vector at point $(\lambda_t, \mathbf{u}_t)$, we have

$$N(\lambda, \mathbf{u}, l) = \dot{\mathbf{u}}^\top (\mathbf{u} - \mathbf{u}_t) + \dot{\lambda}^\top (\lambda - \lambda_t) - l = 0. \tag{24}$$

The resulting modified system makes a step in pseudo-arclength l (rather than λ), and the step is usually set to be a sufficiently small nonnegative scalar. In this study we set

$$l = \|\mathbf{u}_{j+1} - \mathbf{u}_j\|_2, \tag{25}$$

where $(\lambda_j, \mathbf{u}_j)$ is the last regular point such that $j \leq t-1$. Since the path is always smooth at regular points, l is guaranteed to be a small nonnegative scalar.

According to the implicit function theorem [29], the tangent vector $(\dot{\lambda}, \dot{\mathbf{u}})$ can be computed as

$$\begin{aligned} (\dot{\lambda}, \dot{\mathbf{u}}) &= \big(\delta_\lambda, \delta_\lambda \frac{\partial \mathbf{u}}{\partial \lambda}(\lambda_t, \mathbf{u}_t)\big) \\ &= \big(\delta_\lambda, \delta_\lambda (F_{\mathbf{u}}(\lambda_t, \mathbf{u}_t))^{-1} F_\lambda(\lambda_t, \mathbf{u}_t)\big). \end{aligned} \tag{26}$$

where δ_λ is the step length of each iteration. Such calculation, however, is inapplicable to the bifurcation point and computationally expensive. We therefore propose approximation of the tangent vector using previous points for computational efficiency. The tangent vector at a previous iteration i $(i < t)$ is approximated as $(\delta_\lambda, \mathbf{u}_{i+1} - \mathbf{u}_i)$, and the tangent vector $(\dot{\lambda}, \dot{\mathbf{u}})$ at t is estimated as

$$(\dot{\lambda}, \dot{\mathbf{u}}) = \Big(\delta_\lambda, \frac{\sum_{i=1}^{k} (k-i+1)(\mathbf{u}_{t-i+1} - \mathbf{u}_{t-i})}{\sum_{i=1}^{k} i}\Big), \tag{27}$$

where k controls the size of the window used for approximation. The motivation behind this approximation is the smoothness of the path, which implies the similarity between tangent vectors of consecutive iterations. After the tangent vector $(\dot{\lambda}, \dot{\mathbf{u}})$ is computed, we solve Eq. (23) using the *trust-region-reflective* algorithm [6], and then branch a new path using the solution $(\lambda^*, \mathbf{u}^*)$ as the initial solution.

The Jacobian of the pseudo-arclength system (Eq. (23)) is the bordered matrix $\left[\begin{array}{cc} F_{\mathbf{u}} & F_\lambda \\ \dot{\mathbf{u}} & \dot{\lambda} \end{array} \right]$. Appending the tangent vector as the last row can be seen as determining the coefficient of the null vector in the general solution of the Newton system [16] (particular solution plus an arbitrary multiplication of the null vector). It is known that the bordered Jacobian matrix of the pseudo-arclength system is nonsingular under the assumption of simple folds [17], [28].

Considering the multiple paths around a singular point of the system $F(\lambda, \mathbf{u}) = \mathbf{0}$, both APE and PAC choose to search along one of them, but the selection mechanisms of APE and PAC are quite different. The selected path of APE is uncontrollable, which is largely dependent on the start point (it is generated via path prediction using the solutions at previous iterations). In contrast, PAC always choose a smooth path from the multiple paths, where the smoothness is achieved by the appended constraint $N(\lambda, \mathbf{u}, l) = 0$. It is very occasional that our branch switching strategy cannot branch new paths other than the original path computed by APE. In that case, APE coincidentally chooses the same path with PAC.

While PAC can fully replace APE in path following to create generally better solutions, it runs much slower than APE. Fortunately, PAC and APE are equivalent at a regular point where only one path pass through, thus PAC is needed only at singular points which are usually sparsely distributed. Consequently, we utilize APE, rather than PAC, at regular points to reduce the computational burden.

6.3 Apply APE and BPF to GNCCP

As one of the latest work following the path following strategy, the *graduated nonconvexity and concavity procedure* (GNCCP) [36] algorithm has demonstrated state-of-the-arts performance in graph matching. GNCCP realizes CCRP on partial permutation matrix without explicitly involving the convex or concave relaxation. It provides a general optimization framework for the combinatorial optimization problems defined on the set of partial permutation matrices, which is not restricted to weighted graph matching of Eq. (2) but is also suitable for the pairwise compatibility form of Eq. (3). Now we describe how to improve GNCCP by integrating the proposed branching path following strategy.

In Alg. 1, line 5 corresponds to the predictor and lines 7 to 11 the corrector via the Frank-Wolfe algorithm. Function $\mathcal{E}_\lambda$ in lines 8 and 9 is the linear interpolation between the convex relaxation and the concave relaxation of the original objective function, which is the main difference between different path following algorithms (more details in [36], [57], [60]). By specifying $\mathcal{E}_\lambda$ to be different objective functions, the proposed strategies can be integrated into various path following algorithms. In particular, we integrate the proposed APE and BPF strategies into GNCCP, and denote the enhanced algorithm as ABPF-G.

Note that, in both steps of singular point discovery and branch switching in BPF, we need to compute the Jacobian of F in advance, which includes a parameter-dependent sub-matrix, the Jacobian of $\nabla \mathcal{E}_\lambda(\mathbf{x})$ (denoted as $J_\lambda(\mathbf{x})$). In GNCCP

$$\mathcal{E}_\lambda(\mathbf{x}) = (1 - |\lambda|)\mathbf{x}^\top K \mathbf{x} + \lambda \mathrm{tr}(\mathbf{x}^\top \mathbf{x}). \tag{28}$$

Applying BPF here, we have

$$J_\lambda(\mathbf{x}) = (1 - |\lambda|)(K^\top + K) + 2\lambda \mathbf{I}. \tag{29}$$

The approach introduces several parameters including the window size k, the minimum step length $\delta_{\min}$, the step length adaption rate ρ and the threshold for path estimation θ. In all the experiments we set $k = 5$, $\delta_{\min} = 0.001$, $\rho = 2$ and $\theta = 0.01n$.

6.4 Optimality and computational efficiency

BPF computes an initial path given an initial point, and branches new paths at discovered singular points in the initial path. Since the solution of the original algorithm is always in set T, the new algorithm is guaranteed to achieve a better, or very occasionally the same, solution in terms of optimization objectives. It is notable that there may also exist singular points on the newly explored paths, and theoretically these singular points may lead to even better solutions if we branch paths along them recursively. However,

the computational burden of such exhaustive strategy increases exponentially, which makes it impractical in real applications. As a result, we consider only the singular points in the initial path, ignoring possible singular points in newly explored paths.

The computational complexity of GNCCP is $O(n^3)$ [36] for graphs of n nodes. Thus, the computational complexity of the new ABPF-G algorithm is $O(n_p n^3)$, where n_p is the number of explored additional branches. Theoretically, n_p is bounded by the number of samples along the original path, that is, $n_p \leq 1/\delta_{\min}$ because there are at most $1/\delta_{\min}$ singular points. In practice, singular points are very sparse along the path, and in fact we have observed that $n_p \leq 10$ usually holds in all our experiments. As a result, the computational complexity of the proposed ABPF-G algorithm roughly equals to $O(n^3)$ by dropping n_p.

Moreover, the APE strategy significantly accelerates ABPF-G, which actually runs in a speed comparable to the original GNCCP. In our experiment, though ABPF-G needs to compute multiple paths to search better solutions, we find it achieves comparable and sometimes even better computational efficiency compared with the original GNCCP algorithm.[1] One reason is that the embedded APE strategy accelerates significantly the convergence of searching along each path.

Interestingly, APE by itself brings benefits to not only computational efficiency but also matching results. In fact, ABPF-G in this work obtains better matching performance than that in our preliminary work [54] where APE is absent. The gain in matching performance mainly comes from the smaller step length which is made affordable by integrating APE to accelerate convergence of searching paths.[2]

Finally, as general strategies, APE and BPF can be applied to other path following algorithms. In such cases, we expect the similar analysis in the optimality and complexity as above.

7 Experiments

In this section, the proposed ABPF-G algorithm is evaluated thoroughly in comparison with six state-of-the-art graph matching algorithms on five datasets, including a synthetic dataset and four public benchmarks. The experimental results clearly support the benefit brought by the proposed APE and BPF strategies.

7.1 Experiment setup

We compare the proposed ABPF-G algorithm with six state-of-the-art graph matching algorithms:

IPFP [33]: The IPFP (integer projected fixed point method) algorithm can take any continuous or discrete solution as inputs and iteratively improve the solution. In our experiments, we start from the initial solution as in [33].

RRWM [12]: The RRWM (re-weighted random walk matching) algorithm reformulates graph matching as a node selection problem and uses an affinity-preserving random walk algorithm to solve the problem.

PSM [20]: The PSM (probabilistic spectral graph matching) algorithm extends the PM algorithm [58] by dropping some constraints, and presents a probabilistic interpretation of the spectral relaxation scheme proposed in [32].

GAGM [24]: The GAGM (graduated assignment graph matching) algorithm employ the graduated assignment technique to iteratively solve a series of linear approximations of the cost function using the Taylor expansion.

FGM [60]: FGM (factorized graph matching) follows the path following strategy and factorizes an affinity matrix into a Kronecker product of smaller matrices, each of them encodes the structure of the graphs and the affinities between nodes and between edges.

GNCCP [36]: GNCCP (graduated non-convexity and concavity procedure) algorithm is one of the latest path following graph matching algorithm. It equivalently realizes CCRP on a partial permutation matrix and provides a much simpler form without involving the convex or concave relaxation explicitly.

Two indicators, *matching accuracy* and *objective ratio*, are used to evaluate algorithms. Denote $H \in \{0,1\}^{n_1 \times n_2}$ the ground-truth assignment matrix where $H_{ij} = 1$ if and only if there is a ground-truth correspondence between $v_i \in \mathbb{V}^{(1)}$ and $v_j \in \mathbb{V}^{(2)}$. For matching accuracy, denote X the assignment matrix acquired by a matching algorithm, then its matching accuracy r is defined as $r = \mathrm{tr}(X^\top H)/\mathrm{tr}(H^\top H)$.

For objective ratio, denote $\mathcal{E}^{(i)}$ the objective achieved by the i-th algorithm, $i = 1, 2, \ldots, 7$, the objective ratio r_i for algorithm i is computed as $r_i = \mathcal{E}^{(i)} / \max_j \mathcal{E}^{(j)}$. Since these algorithms use the same objective function defined in Eq. (3),[3] a higher objective ratio indicates a better solution in line with the objective.

7.2 Synthetic dataset

7.2.1 Matching random graphs

We first perform a comparative evaluation of the algorithms on graphs randomly synthesized following the experimental protocol in [12], [60]. For each trial, we construct two graphs, $\mathbb{G}^{(1)}$ and $\mathbb{G}^{(2)}$, with $n_{\text{in}} = 20$ inlier nodes, and later add n_{out} outlier nodes to both graphs. The edges between nodes are randomly generated independently for each of two graphs to match, with respect to an edge density parameter $\nu \in [0, 1]$ following [60]. Each edge (i, j) in $\mathbb{G}^{(1)}$ is assigned with an edge weight $A^{(1)}_{ij}$ sampled uniformly in [0,1]. For (i, j)'s corresponding edge (a, b) in $\mathbb{G}^{(2)}$, its weight is assigned as $A^{(2)}_{ab} = A^{(1)}_{ij} + \epsilon$, such that $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian perturbation. The edge affinity is computed as $K_{\mathrm{ind}(i,a)\mathrm{ind}(j,b)} = \exp\big(-\big(A^{(1)}_{ij} - A^{(2)}_{ab}\big)^2/0.15\big)$ and the node affinity is set to zero.

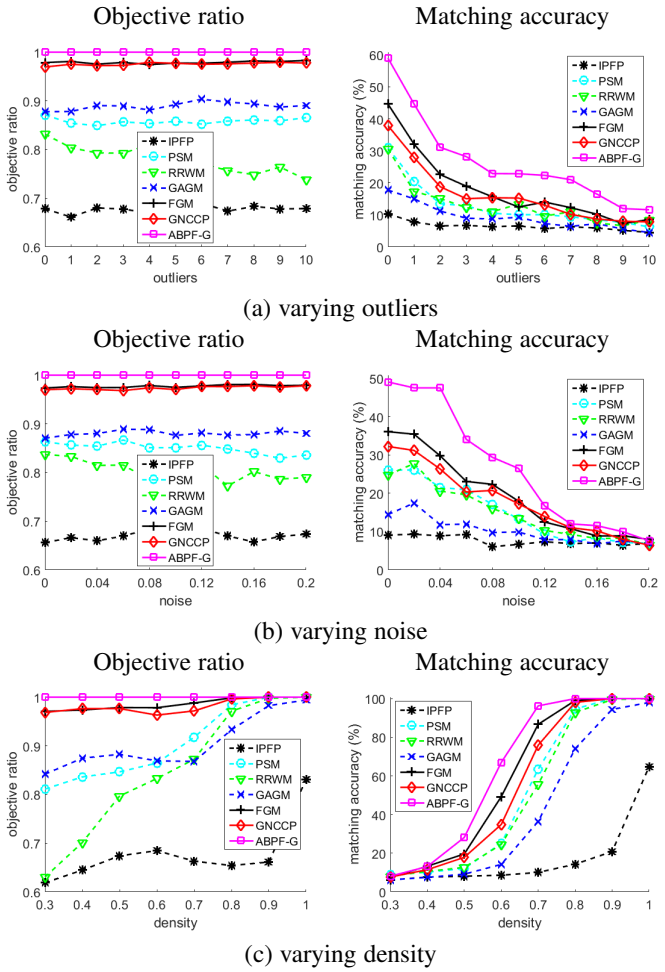
We compare the performance of the algorithms under three different settings by varying the number of outliers n_{out}, edge density ν and edge noise σ, respectively. For each setting, we construct 100 different pairs of graphs and evaluate the average matching accuracy and objective ratio. In the first setting (Fig. 2(a)), we fix edge density $\nu = 0.5$ and edge noise $\sigma = 0$, and increase the number of outliers n_{out} from 0 to 10. In the second setting (Fig. 2(b)), we change the edge noise parameter σ from 0 to 0.2 while fixing $n_{\text{out}} = 0$ and $\nu = 0.5$. In the last case (Fig. 2(c)), the edge density ν ranges from 0.3 to 1, and the other two parameters are fixed as $n_{\text{out}} = 0$ and $\sigma = 0.1$.

The comparison on both objective ratio and matching accuracy is summarized in Fig. 2. It can be observed that in almost all of cases under varying parameters, our approach achieves the best performance in terms of both objective ratio and matching accuracy. All algorithms are harmed by the increase of outliers

1. For fairness, the step length of GNCCP is set to be equal to $\delta_{\min}$ in our experiments, and it is fixed across all iterations in GNCCP.

2. The step length in our preliminary work [54] is fixed to 0.01, comparing $\delta_{\min} = 0.001$ in this work

3. Although a different formulation based on factorized matrices is adopted in FGM, it is equivalent to the objective function in Eq. (3).

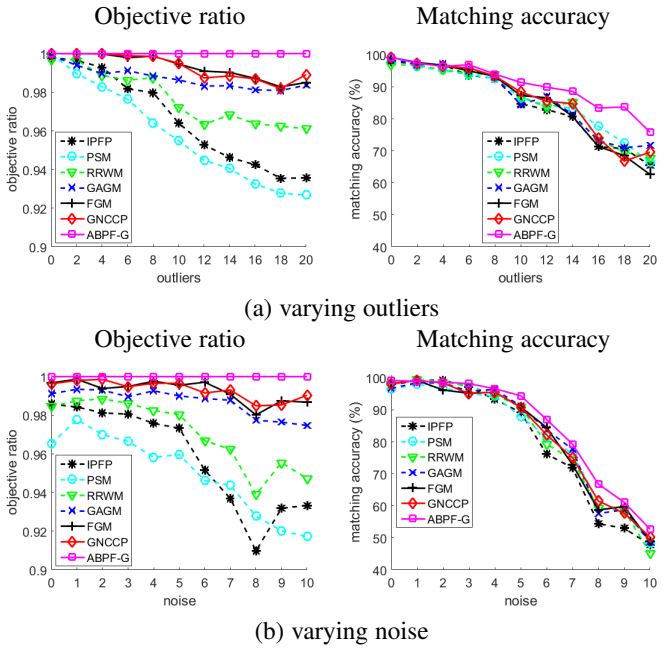


(a) varying outliers

(b) varying noise

(c) varying density

Fig. 2. Evaluation on the synthetic dataset under different settings.



(a) varying outliers

(b) varying noise

Fig. 3. Comparison of graph matching on 2D point sets.

and edge noise, but are benefited from the increase of edge density. From Fig. 2(c), the PSM, RRWM, FGM and GNCCP algorithms are comparable to our approach when the graphs are close to full connections (the density parameter ρ near to 1). All algorithms fail to achieve reasonable solutions when graph pairs present extreme perturbation or sparsity.

7.2.2 Matching 2D points

To evaluate the algorithms in a more realistic scenario, we compare them on the task of finding correspondences between sets of 2D points generated following the experimental protocol in [32]. For each trial, we construct two sets, $\mathbb{P}^{(1)}$ and $\mathbb{P}^{(2)}$, with $n_{\text{in}} = 10$ inlier points, and later add n_{out} outlier points to both sets. The inlier points in $\mathbb{P}^{(1)}$ are randomly selected in a given region of the plane. We then obtain the corresponding inliers in $\mathbb{P}^{(2)}$ by disturbing independently the points from $\mathbb{P}^{(1)}$ with white gaussian noise $\mathcal{N}(0, \sigma)$, and then rotating and translating the whole data set $\mathbb{P}^{(2)}$ with a random rotation and translation. The outlier points of each set are randomly selected from the same region as liners. The range of the point coordinates in $\mathbb{P}^{(1)}$ is $256\sqrt{(n_{\text{in}} + n_{\text{out}})/10}$. Given two pair of points $(i, j) \in \mathbb{P}^{(1)}$ and $(a, b) \in \mathbb{P}^{(2)}$, The edge affinity is computed as

$$K_{\text{ind}(i,a)\text{ind}(j,b)} = \begin{cases} 4.5 - \frac{(d_{ij} - d_{ab})^2}{2\sigma_d^2}, & \text{if } |d_{ij} - d_{ab}| < 3\sigma_d, \\ 0, & \text{otherwise.} \end{cases}$$

where d_{ij} (respectively d_{ab}) denotes the Euclidean distance between points i and j (respectively a and b), and the parameter σ_d controls the sensitivity of the score on deformations. We set $\sigma_d = 5$ throughout our experiments.

We compare the performance of the algorithms under two different settings by varying the number of outliers n_{out} and noise parameter σ respectively. For each setting, we construct 100 different pairs of point sets and evaluate the average matching accuracy and objective ratio. In the first setting (Fig. 3(a)), we hold $\sigma = 5$ and increase n_{out} from 0 to 20. In the second setting (Fig. 3(b)), we change σ from 0 to 10 while fixing $n_{\text{out}} = 10$. The comparison on both objective ratio and matching accuracy is summarized in Fig. 3. It is shown that, our approach exhibits the best performance in terms of both objective ratio and matching accuracy, in all of cases under varying parameters.

7.3 CMU house dataset

The CMU house dataset includes 111 frames of image sequences, where all sequences contain the same house object with transformation cross sequence gaps. In order to assess the matching accuracy, following [7], [49], 30 landmarks were manually tracked and labeled across all frames. We matched all possible image pairs, in total 560 pairs gapped by 10, 20, ..., 100 frames, where increasing sampling gaps implies the increase of deformation degree. To evaluate graph matching algorithms against noise, we use two different settings of nodes (n_1, n_2) =(30,30) and (20, 30), where decreasing n_1 implies the increase of outlier. In the setting where $n_1 < 30$, n_1 points are randomly chosen out of the 30 landmark points.

We model each landmark as a graph node, and then build graph edges by Delaunay triangulation [31]. Each edge (i, j) is associated with a weight A_{ij}, which is computed as the Euclidean distance between the connected nodes v_i and v_j. The node affinity is set to zero, and the edge affinity between edges (i, j) in $\mathbb{G}^{(1)}$ and (a, b) in $\mathbb{G}^{(2)}$ is computed as $K_{\text{ind}(i,a)\text{ind}(j,b)} = \exp(-(A^{(1)}_{ij} - A^{(2)}_{ab})^2/2500)$.

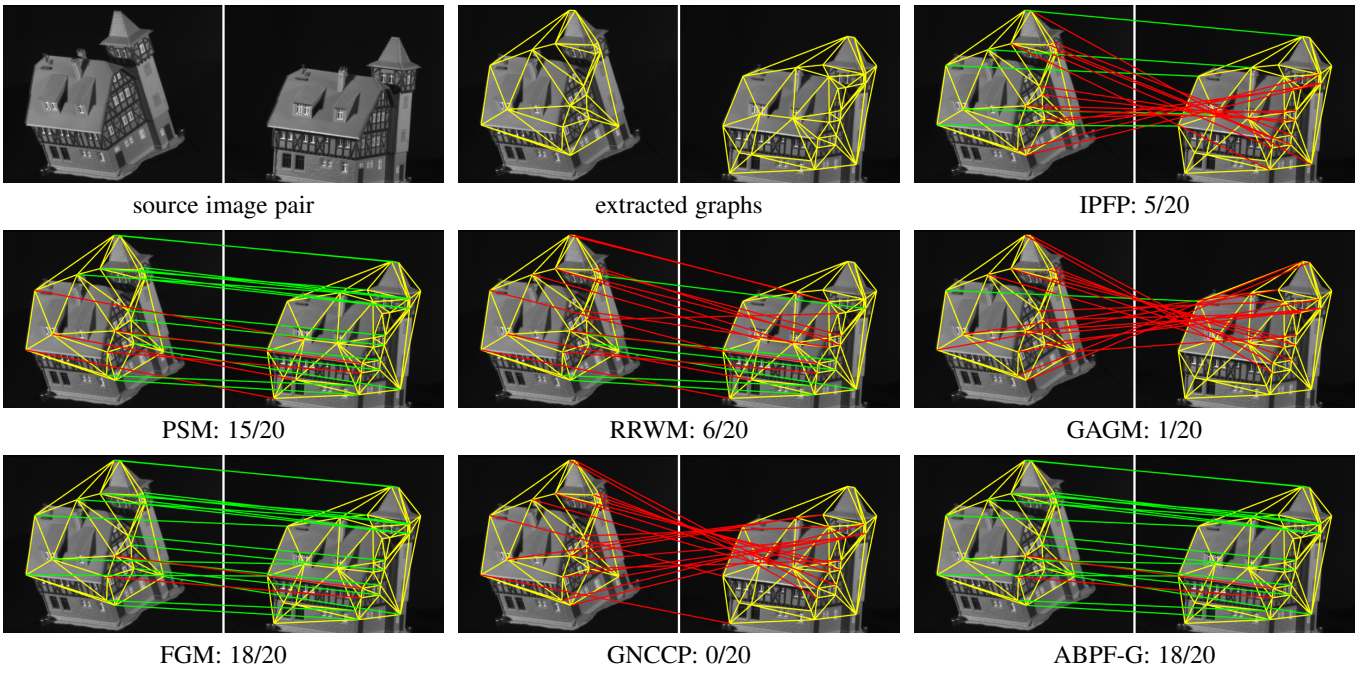
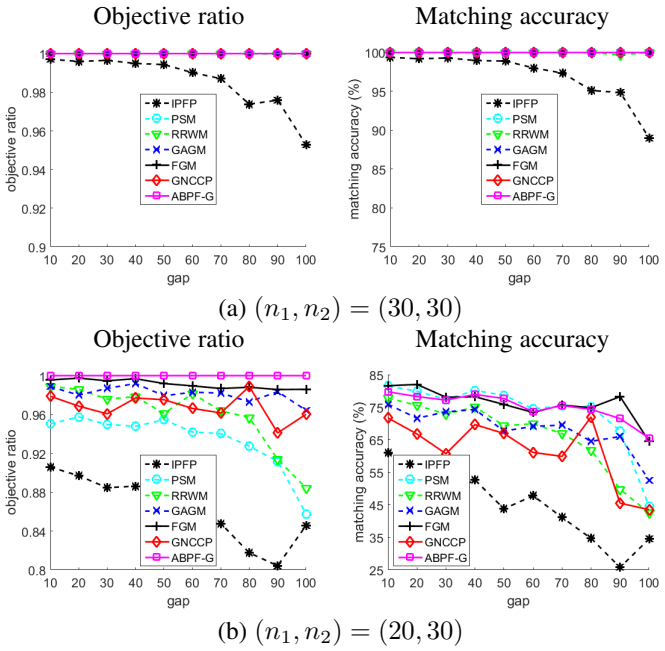


Fig. 5. An example of graph matching on the CMU house dataset. The algorithm, the number of true matches per ground truths for each subfigure are captioned. Graph edges are represented by yellow lines, true matches by green lines and false matches by red lines (best viewed in color, and the same style is also used for figures 6, 8 and 9).



(a) $(n_1, n_2) = (30, 30)$

(b) $(n_1, n_2) = (20, 30)$

Fig. 4. Comparison of graph matching on the CMU house dataset.

Fig. 5 presents an example for graph matching with 10 outliers under significant deformation. Fig. 4 shows the performance curves for n_1 = 30 and 20 with respect to variant sequence gaps. This dataset is a well-studied benchmark for graph matching, and all algorithms except IPFP achieve perfect matching when no outlier exists ($n_1 = 30$). When we increase the number of outliers to 10 ($n_1 = 20$), our approach gains remarkable improvement in both accuracy and objective compared with the original GNCCP algorithm. It is interesting to see that PSM and FGM gain comparable matching accuracy to our approach with certain sequence gaps but achieve lower objectives.

7.4 Pascal dataset

The Pascal dataset [34] consists of 30 pairs of car images and 20 pairs of motorbike images selected from Pascal 2007 [21]. The authors provide detected feature points and manually labeled ground-truth correspondences for each pair of images. To evaluate the performance of each algorithm against noise, we randomly select $0 \sim 20$ outlier nodes from the background.

For graph construction and affinity computation, we follow the procedure in [60]. For each node v_i, we associate it with a feature p_i which is computed as its orientation of the normal vector at that point to the contour where the point was sampled. The node affinity between nodes v_i and v_j is consequently computed as $\exp(-|p_i - p_j|)$. We use Delaunay triangulation to build graph edges, and associate each edge (i, j) with two features d_{ij} and θ_{ij}, where d_{ij} is the pairwise distance between the connected nodes v_i and v_j, and θ_{ij} is the absolute angle between the edge and the horizontal line, i.e., $0 \leq \theta_{ij} \leq \pi/2$. Consequently, the edge affinity between edges (i, j) in $\mathbb{G}^{(1)}$ and (a, b) in $\mathbb{G}^{(2)}$ is computed as $K_{\mathrm{ind}(i,a)\mathrm{ind}(j,b)} = \exp(-(|d_{ij} - d_{ab}| + |\theta_{ij} - \theta_{ab}|)/2)$.

Fig. 6 presents an example for graph matching of motorbike images (10 outliers). The matching accuracy and objective ratio of each algorithm with respect to the number of outliers are summarized in Fig. 7(a). It can be seen that our approach outperforms other algorithms remarkably in both matching accuracy and objective ratio. In the case when no outlier exists, our method achieves near 90% matching rate, while GNCCP and FGM obtain similar matching performance. With the increase of outlier level, our method shows stronger robustness than other algorithms.

7.5 Willow object dataset

In this experiment, following [10], we work on 500 image pairs using the Willow object class dataset. This dataset provides images

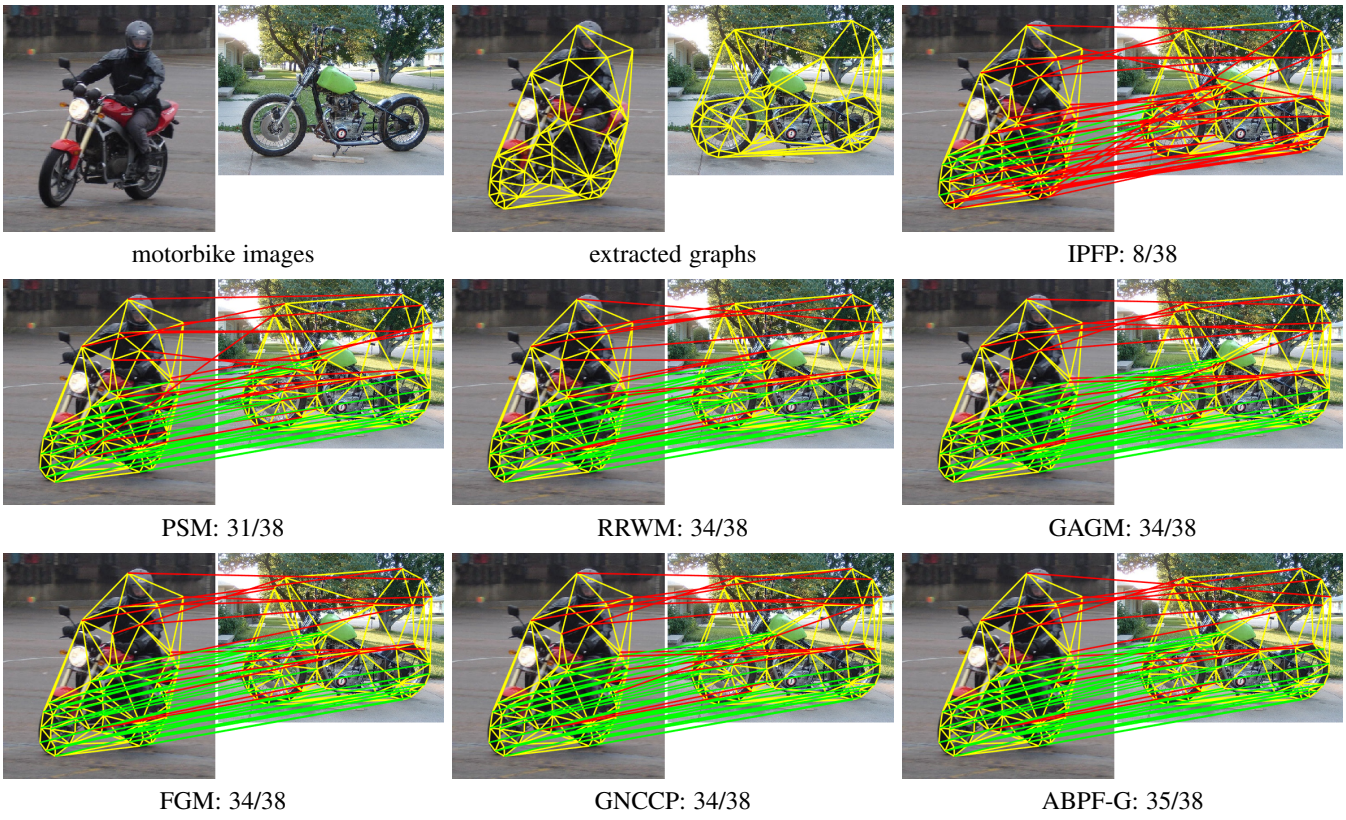
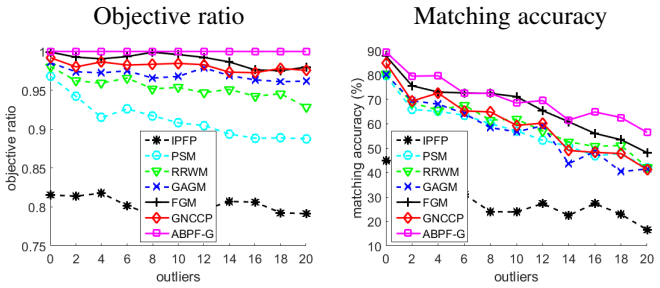
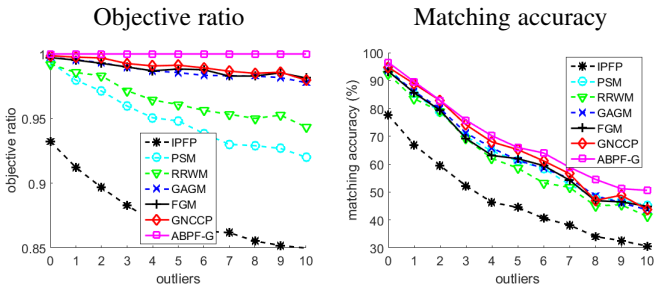


Fig. 6. A matching example of motorbike images in the Pascal dataset.



(a) Pascal dataset



(b) Willow dataset

Fig. 7. Evaluation on (a) the Pascal dataset and (b) the Willow dataset.

of five classes, namely car, duck, face, motorbike and winebottle. Each class contains at least 40 images with different instances and 10 distinctive landmarks were manually labeled on the target object across all images in each class. We randomly select 100 pairs of images from each class respectively.

We use Hessian detector [42] to extract interesting points and SIFT descriptor [38] to represent the node attributes. To test the performance against noise, we randomly select $0 \sim 10$ outlier nodes from the background. We utilize the Delaunay triangulation to connect nodes and compute the affinity between nodes via their appearance similarity. Edge affinity is computed following the method used in subsection 7.4.

Fig. 8 shows a representative example for graph matching selected from the duck class (10 outliers). The comparison on matching accuracy and objective ratio is summarized in Fig. 7(b). It is observed that our approach again performs better than other algorithms in both matching accuracy and objective ratio. The performance gaps are small when there are no outliers, but the gaps increase roughly in line with the number of outliers.

7.6 Caltech image dataset

The Caltech image dataset [12] contains 30 pairs of real images which are mainly collected from Caltech-101 dataset [35].

The authors provide detected MSER keypoints [18], initial matches, affinity matrix, and manually labeled ground-truth correspondences for each image pair. In [12], candidate matches were filtered according to the distance between SIFT features [38], and all the low-quality candidate matches were filtered if the feature pair has farer distance than a loose threshold $\delta = 0.6$. The affinity matrix is consequently computed by the mutual projection error function [11] considering the similarity between both the appearance-based feature descriptors and the geometric transformations. The FGM and GAGM algorithms are absent in this experiment because the provided affinity matrices cannot be directly utilized in these two algorithms due to its omission of many candidate matches.

Fig. 9 shows a representative example for graph matching with significant deformation and plenty of repeated patterns. The

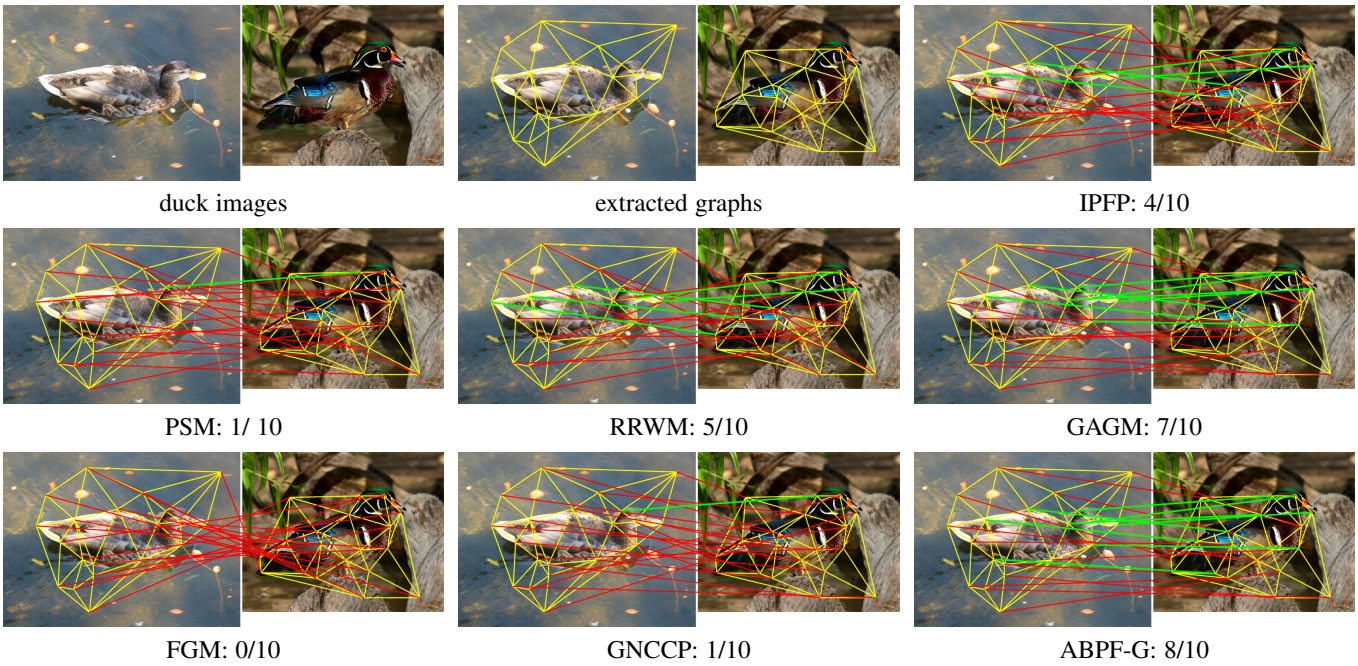


Fig. 8. A matching example of duck images on the Willow dataset.

TABLE 1
Comparison of graph matching on the Caltech dataset.

Algorithm	IPFP [33]	RRWM [12]	PSM [20]	GNCCP [36]	ABPF-G (ours)
Objective ratio	0.91	0.88	0.76	0.92	**1**
Accuracy (%)	68.71	75.49	67.40	68.72	**75.51**

TABLE 2
Comparison with baselines in matching accuracy (%) and computational time (seconds) on four image datasets.

Criterion	Method	CMU House n_1=30	CMU House n_1=20	Pascal	Willow	Caltech
Accuracy	GNCCP	100	59.98	58.87	66.73	68.72
	APE-G	100	60.88	61.64	65.26	67.63
	BPF-G	100	71.76	67.08	68.27	73.56
	ABPF-G	100	**75.86**	**70.80**	**69.41**	**75.51**
Speed	GNCCP	10.6	22.1	33.9	5.03	4.81
	APE-G	**0.38**	**0.06**	**0.29**	**0.02**	**0.29**
	BPF-G	14.3	58.9	253	18.6	68.6
	ABPF-G	4.52	8.13	74.5	3.12	17.4

matching accuracy and objective ratio of each algorithm are summarized in Table 1. Our approach gains remarkable improvement compared with the original GNCCP algorithm, and also outperforms other algorithms, in both matching accuracy and objective ratio. It is interesting to see that RRWM is comparative to our approach in terms of matching accuracy, but gains much lower objectives than our approach.

7.7 Comparison with base path following algorithms

Being an upgraded algorithm based on GNCCP, the proposed ABPF-G algorithm gains consistent improvement on matching

TABLE 3
Comparison in computational time (seconds) on four image datasets.

Method	CMU House n_1=30	CMU House n_1=20	Pascal	Willow	Caltech
IPFP [33]	**0.01**	**0.01**	**0.02**	**0.01**	**0.11**
GAGM [24]	0.04	0.26	1.26	0.08	-
RRWM [12]	0.02	0.05	0.23	0.02	0.68
PSM [20]	0.28	3.36	9.76	1.22	14.7
FGM [60]	0.52	1.28	3.35	2.15	-
GNCCP [36]	10.6	22.1	33.9	5.03	4.81
ABPF-G	4.52	8.13	74.5	3.12	17.4

results on all datasets, as described in the previous subsection and summarized in Table 2. It is notable that the better matching performance does not necessarily cost higher computational burden. Although ABPF-G needs to compute multiple paths, embedding of APE accelerates significantly the process of searching each path, and thus results in comparative and even better computational efficiency on some datasets.

To analyze the contributions of the two components (APE and BPF) of our method, we report detailed comparison between the original GNCCP and three variations of our method in Table 2, where APE-G (or BPF-G) denotes the downgraded version that applies the APE (or BPF) component only. With the assistance of the APE component, APE-G improves dramatically the computation efficiency, and achieves the similar matching accuracy compared with GNCCP. Applying the BPF component only, BPF-G gains remarkable improvement on matching results, at the cost of higher computational burden. Compared with the preliminary version of our algorithm in [54] (BPF-G), the new version in this paper (ABPF-G) improves not only in speed but also in accuracy. We observe that ABPF-G achieves higher computational efficiency than GNCCP on the CMU House and the Willow object datasets.

We also provide the average computational time of other com-

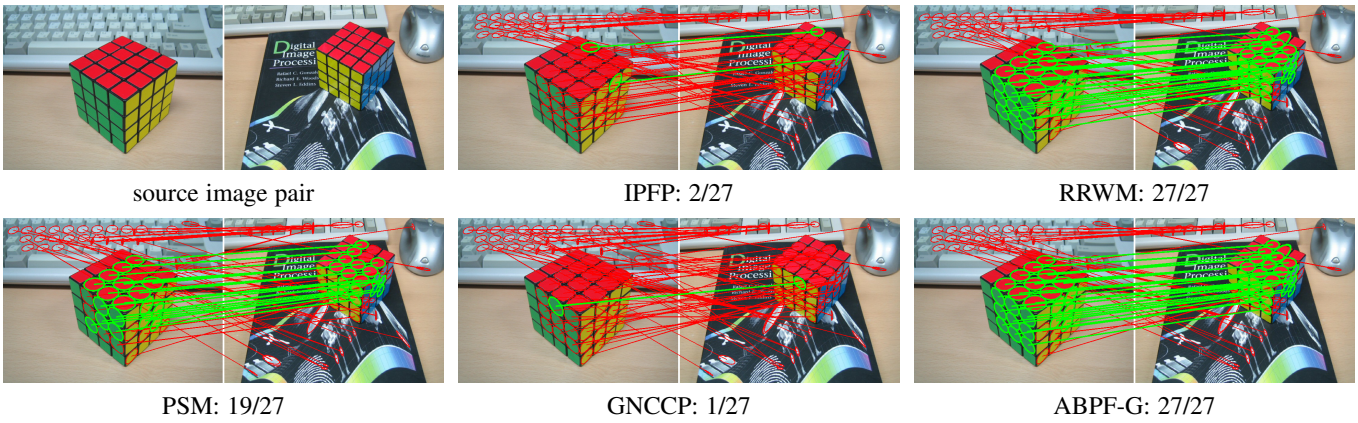


Fig. 9. A matching example on the Caltech dataset.

pared algorithms in Table 3, where all the compared algorithms are implemented in Matlab. It is shown that IPFP is the most efficient one, while GNCCP and our ABPF-G are more computationally expensive than other algorithms.

8 Conclusion

In this paper, we proposed a novel branching path following strategy for graph matching aiming to improve the matching performance. To avoid the pitfalls at singular points in the original path following strategy, our new strategy first discovers singular points and subsequently branches new paths from them seeking for potentially better solutions. Furthermore, an adaptive path estimation method is devised to accelerate the convergence of searching along each path. We applied the strategy to a state-of-the-art graph matching algorithm that utilizes the original path following strategy. Experimental results reveal that, our new graph matching algorithm, named ABPF-G, gains remarkable improvement on matching performance compared with the original algorithm, and also outperforms other state-of-the-art algorithms.

Acknowledgment. This work is supported by the China National Key Research and Development Plan (Grant No. 2016YFB1001200), National Nature Science Foundation of China (nos. 61673048, 61672088, 61671048 and 61472028), the Fundamental Research Funds for the Central universities (2017JBZ108), and in part by US NSF Grants 1618398, 1449860 and 1350521.

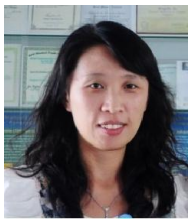
References

[1] K. Adamczewski, Y. Suh, and K. M. Lee. Discrete tabu search for graph matching. In *Proceedings of the 15th IEEE International Conference on Computer Vision*, pages 109–117, 2015.
[2] A. Albarelli, S. R. Bulò, A. Torsello, and M. Pelillo. Matching as a non-cooperative game. In *Proceedings of 12th IEEE International Conference on Computer Vision*, pages 1319–1326, 2009.
[3] E. L. Allgower and K. Georg. *Numerical continuation methods*. Springer, 1990.
[4] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):522–525, 1993.
[5] X. Bai, X. Yang, L. J. Latecki, W. Liu, and Z. Tu. Learning context-sensitive shape similarity by graph transduction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(5):861–874, 2010.
[6] M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.
[7] T. S. Caetano, T. Caelli, D. Schuurmans, and D. A. C. Barone. Graphical models and point pattern matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1646–1663, 2006.
[8] Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, and S. Z. Li. Robust deformable and occluded object tracking with dynamic graph. *IEEE Trans. Image Processing*, 23(12):5497–5509, 2014.
[9] C. Chen and K. Grauman. Efficient activity detection with max-subgraph search. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1274–1281, 2012.
[10] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *Proceedings of the 14th IEEE International Conference on Computer Vision*, pages 25–32, 2013.
[11] M. Cho, J. Lee, and K. M. Lee. Feature correspondence and deformable object matching via agglomerative correspondence clustering. In *Proceedings of the 12th IEEE International Conference on Computer Vision*, pages 1280–1287, 2009.
[12] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *Proceedings of the 11th European Conference on Computer Vision*, pages 492–505, 2010.
[13] 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(3):265–298, 2004.
[14] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, pages 313–320, 2006.
[15] H. Dankowicz and F. Schilder. An extended continuation problem for bifurcation analysis in the presence of constraints. *J. Comput. Nonlinear Dynam*, 6(3):1–14, 2010.
[16] P. Deuflhard. *Newton methods for nonlinear problems - affine invariance and adaptive algorithms*. Series Computational Mathematics 35, Springer, 2006.
[17] K. I. Dickson, C. T. Kelley, I. C. F. Ipsen, and I. G. Kevrekidis. Condition estimates for pseudo-arclength continuation. *SIAM Journal on Numerical Analysis*, 45(1):263–276, 2007.
[18] M. Donoser and H. Bischof. Efficient maximally stable extremal region (MSER) tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 553–560, 2006.
[19] O. Duchenne, F. R. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1980–1987, 2009.
[20] A. Egozi, Y. Keller, and H. Guterman. A probabilistic approach to spectral graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):18–27, 2013.
[21] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
[22] P. Foggia, G. Percannella, and M. Vento. Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(1):1–40, 2014.
[23] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–100, 1956.
[24] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):377–388, 1996.
[25] B. Jiang, J. Tang, X. Cao, and B. Luo. Lagrangian relaxation graph matching. *Pattern Recognition*, 61:255–265, 2017.
[26] X. Jiang and H. Bunke. Optimal quadratic-time isomorphism of ordered graphs. *Pattern Recognition*, 32(7):1273–1283, 1999.
[27] D. Justice and A. O. Hero. A binary linear programming formulation of the graph edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*,

28(8):1200–1214, 2006.
[28] H. B. Keller. *Lectures on numerical methods in bifurcation theory*. Tata Institute of Fundamental Research Lectures on Mathematics and Physics 79, Springer-Verlag, Berlin, 1987.
[29] L. Kudryavtsev. *Implicit function*. Encyclopedia of Mathematics, Springer, 2001.
[30] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of 2nd Berkeley Symposium*, pages 481–492, 1951.
[31] D. T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.
[32] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of 10th IEEE International Conference on Computer Vision*, pages 1482–1489, 2005.
[33] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *Proceedings of the 23th Annual Conference on Neural Information Processing Systems*, pages 1114–1122, 2009.
[34] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *International Journal of Computer Vision*, 96(1):28–45, 2012.
[35] F.-F. Li, M. Andreetto, and M. A. Ranzato. Caltech 101 dataset. http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html.
[36] Z. Liu and H. Qiao. GNCCP - graduated nonconvexityand concavity procedure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(6):1258–1267, 2014.
[37] Z. Liu, H. Qiao, X. Yang, and S. C. H. Hoi. Graph matching by simplified convex-concave relaxation procedure. *International Journal of Computer Vision*, 109(3):169–186, 2014.
[38] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
[39] Y. Lu, K. Huang, and C. Liu. A fast projected fixed-point algorithm for large graph matching. *Pattern Recognition*, 60:971–982, 2016.
[40] V. Lyzinski, D. E. Fishkind, M. Fiori, J. T. Vogelstein, C. E. Priebe, and G. Sapiro. Graph matching: Relax at your own risk. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(1):60–73, 2016.
[41] D. Michel, I. Oikonomidis, and A. A. Argyros. Scale invariant and deformation tolerant partial shape matching. *Image Vision Comput.*, 29(7):459–469, 2011.
[42] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision*, pages 128–142, 2002.
[43] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Computation*, 27(7):950–959, 2009.
[44] E. Rodolà, A. M. Bronstein, A. Albarelli, F. Bergamasco, and A. Torsello. A game-theoretic approach to deformable shape matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 182–189, 2012.
[45] A. Sanfeliu and K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Systems, Man, and Cybernetics*, 13(3):353–362, 1983.
[46] E. Serradell, M. A. Pinheiro, R. Sznitman, J. Kybic, F. Moreno-Noguer, and P. Fua. Non-rigid graph registration using active testing search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):625–638, 2015.
[47] F. Serratosa. Fast computation of bipartite graph matching. *Pattern Recognition Letters*, 45:244–250, 2014.
[48] M. Shacham. Numerical solution of constrained nonlinear algebraic equations. *International Journal for Numerical Methods in Engineering*, 23:1455–1481, 1986.
[49] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *Proceedings of the 10th European Conference on Computer Vision*, pages 596–609, 2008.
[50] J. Wang and S. Li. Query-driven iterated neighborhood graph search for large scale indexing. In *Proceedings of the 20th ACM Multimedia Conference*, pages 179–188, 2012.
[51] T. Wang and H. Ling. Path following with adaptive path estimation for graph matching. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 3625–3631, 2016.
[52] T. Wang and H. Ling. Gracker: A graph-based planar object tracker. *IEEE Trans. Pattern Anal. Mach. Intell.*, in press.
[53] T. Wang, H. Ling, C. Lang, and S. Feng. Symmetry-aware graph matching. *Pattern Recognition*, 60:657–668, 2016.
[54] T. Wang, H. Ling, C. Lang, and J. Wu. Branching path following for graph matching. In *Proceedings of 14th European Conference on Computer Vision*, pages 508–523, 2016.
[55] J. Wu, H. Shen, Y. Li, Z. Xiao, M. Lu, and C. Wang. Learning a hybrid similarity measure for image retrieval. *Pattern Recognition*, 46(11):2927–2939, 2013.
[56] J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, and S. M. Chu. Discrete hyper-graph matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1520–1528, 2015.
[57] M. Zaslavskiy, F. R. Bach, and J. Vert. A path following algorithm for the graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2227–2242, 2009.
[58] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
[59] Q. Zhang, Y. N. Wu, and S. Zhu. Mining and-or graphs for graph matching and object discovery. In *Proceedings of the 15th IEEE International Conference on Computer Vision*, pages 55–63, 2015.
[60] F. Zhou and F. De la Torre. Factorized graph matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 127–134, 2012.
[61] F. Zhou and F. De la Torre. Factorized graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(9):1774–1789, 2016.

Tao Wang received the Ph.D. degree in the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, P.R. China, in 2013. He is currently an Associate Professor in the School of Computer and Information Technology, Beijing Jiaotong University. He has been a visiting scholar in the Department of Computer & Information Sciences, Temple University, USA, from 2014 to 2015. His research interests include computer vision and machine learning.

Haibin Ling received B.S. and M.S. degrees from Peking University in 1997 and 2000, respectively; and the Ph.D. degree from the University of Maryland, College Park in 2006. From 2000 to 2001, he was an assistant researcher at Microsoft Research Asia. From 2006 to 2007, he worked as a postdoctoral scientist at the University of California Los Angeles. After that, he joined Siemens Corporate Research as a research scientist. Since fall 2008, he has been with Temple University where he is now an Associate Professor. His research interests include computer vision, augmented reality, medical image analysis, and human computer interaction. He received the Best Student Paper Award at ACM UIST in 2003, and the NSF CAREER Award in 2014. He is an Associate Editor for IEEE Trans. on PAMI and for Pattern Recognition, and served as Area Chairs for CVPR 2014 and CVPR 2016.



Congyan Lang received the Ph.D. degree from the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China, in 2006. She was a Visiting Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, from 2010 to 2011. From 2014 to 2015, she visited the Department of Computer Science, University of Rochester, Rochester, NY, USA, as a Visiting Researcher. She is currently a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. Her current research interests include multimedia information retrieval and analysis, machine learning, and computer vision.

Songhe Feng received the Ph.D. degree in the School of Computer and Information Technology, Beijing Jiaotong University, China, in 2009. He is currently an Associate Professor in the School of Computer and Information Technology, Beijing Jiaotong University. He has been a visiting scholar in the Department of Computer Science and Engineering, Michigan State University, from 2013 to 2014. His research interests include computer vision and machine learning.