Feature Matching with Affine-Function Transformation Models

Hongsheng Li, Xiaolei Huang, *Member, IEEE*, Junzhou Huang, *Member, IEEE*, and Shaoting Zhang, *Member, IEEE*

Abstract—Feature matching is an important problem and has extensive uses in computer vision. However, existing feature matching methods support either a specific or a small set of transformation models. In this paper, we propose a unified feature matching framework which supports a large family of transformation models. We call the family of transformation models *the affine-function family*, in which all transformations can be expressed by affine functions with convex constraints. In this framework, the goal is to recover transformation parameters for every feature point in a template point set to calculate their optimal matching positions in an input image. Given pairwise feature dissimilarity values between all points in the template set and the input image, we create a convex dissimilarity function for each template point. Composition of such convex functions with any transformation model in the affine-function family is shown to have an equivalent convex optimization form that can be optimized efficiently. Four example transformation models in the affine-function family are introduced to show the flexibility of our proposed framework. Our framework achieves 0.0 percent matching errors for both CMU House and Hotel sequences following the experimental setup in [6].

Index Terms—Feature matching, object matching, convex optimization, convex composition

1 Introduction

The problem of feature matching in 2D images can be referred as matching a group of *template* feature points to another group of *image* feature points. Each feature point is associated with a position (x, y) in the 2D image domain and a feature vector describing the image's local appearance around that position. The matched image feature points should maintain similar local appearances as well as relative spatial relationships of the template feature points. In Fig. 1, we show a honeybee represented by a group of template feature points (Fig. 1a), and an input image consisting of thousands of detected image feature points (Fig. 1b). Fig. 1c shows the matching result of this case using our proposed method. Feature matching has extensive uses in image registration [32], shape matching [15], object detection [23], object recognition [4], [11], image retrieval [26], estimation of optical flow [5], [29], and 3D surface reconstruction [25].

1.1 Background

In this section, we review some existing feature matching methods. Because of our interest in transformation models and constraints, we focus on what transformation models or transformation invariants do the methods support.

Assuming a global transformation between the template and the input image, one can estimate the transformation parameters between them by different parameter fitting methods, such as least squares, least-median-of-squares (LMedS), RANdom SAmple Consensus (RANSAC) [13], etc. Among them, the RANSAC and its variants [7], [24] have the ability to tolerate a large amount of outliers and are therefore widely used in different computer vision applications. However, the major limitation of the fitting methods is that they are only capable of handling global transformations.

Graph matching methods have a long history [9] and regained much attention since 2005 [4], [17]. In graph matching methods, every feature point is modeled as a graph node, and pairs of feature points in a same point set are modeled as graph edges. For each edge, some geometric properties are calculated from its pair of feature points. Two most common pairwise geometric properties are the distance and the direction between a pair of feature points. The former geometric property is invariant to a rigid transformation of the two points, while the latter one is invariant to simultaneous translation or scaling of the two points. In graph matching, one can determine how well an edge in one graph matches an edge in another graph by comparing the two edges' geometric properties. Berg et al. [4] modeled the graph matching problem as a integer quadratic programming (IQP) model, where linear terms and quadratic terms of the objective function represent node-to-node and edge-to-edge affinities between the two graphs, respectively. The IQP model is then relaxed into a continuous domain and solved. The continuous result is truncated back to the domain of integers to obtain node-to-node correspondences. Leordeanu and Hebert [17] encoded all the above affinity information into a matrix, where diagonal and off-diagonal entries record node-to-node and edge-to-edge affinities, respectively. The

- *H. Li is with the Department of Information Engineering at University of Electronic Science and Technology of China, Chengdu, China.*
- *X. Huang is with the Department of Computer Science and Engineering at Lehigh University, Bethlehem, PA.*
- *J. Huang is with the Department of Computer Science and Engineering at University of Texas at Arlington, TX.*
- *S. Zhang is with the Department of Computer Science, Rutgers University, NJ.*

Manuscript received 10 Apr. 2013; revised 2 Mar. 2014; accepted 20 Apr. 2014. Date of publication 15 May 2014; date of current version 5 Nov. 2014.
Recommended for acceptance by T. Cootes.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2014.2324568

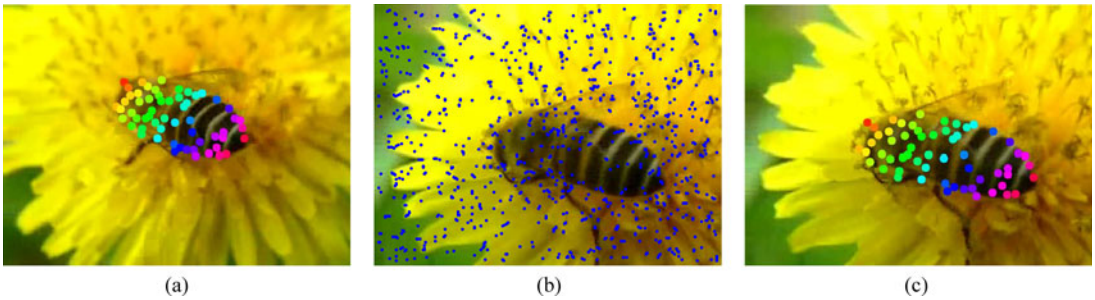


Fig. 1. (a) A group of *template* feature points representing a honeybee. (b) Thousands of *image* feature points representing the input image. (c) The final matching result by our proposed method. SIFT features [22] are used in this example.

correspondences are obtained by mapping the principal eigenvector of the matrix to the discrete solution space using a greedy algorithm. To automatically set the weights of different terms in the affinity matrix, supervised [6] and unsupervised [18] learning methods optimizing the weights were proposed. Cour et al. [8] proposed a spectral relaxation method for the graph matching problem that incorporates one-to-one or one-to-many mapping constraints, and presented a normalization procedure for existing graph matching scoring functions that can dramatically improve the matching accuracy. Torresani et al. [28] proposed a dual decomposition method to decompose the original graph matching problem into "easier" subproblems. Lower bounds provided by solving the subproblems are then maximized to obtain a global solution. Most recently, Liu and Yan [21] proposed an algorithm to discover all common visual patterns within two sets of feature points. It optimizes the same objective function as that of [17] but with different constraints. It showed its effectiveness in recovering visual common patterns no matter whether the matchings between them are one-to-one or many-to-many. Torki and Elgammal [27] presented a novel two-step graph matching method. Given two groups of feature points with pairwise intra-group and inter-group affinities, the two groups of feature points were first embedded into a unified Euclidean space via Laplacian Embedding. A bipartite graph matching was then applied to the embedded feature points to efficiently recover point-to-point correspondences. One disadvantage of graph matching methods is their high computation complexity.

Another disadvantage of graph matching methods is that the most commonly used second-order edges (edges connecting two nodes) in graph matching methods can encode a limited number of geometric invariants. Zass and Shashua [31] for the first time tried to solve the feature matching problem via hypergraph matching, whose high-order edges can encode more complex geometric invariants. The method's output is a probabilistic result rather than traditional node-to-node results. In this way, they were able to model the problem as a convex optimization problem and to obtain a global minimum. Duchenne et al. [10] encoded affinities between two hypergraphs into a tensor and proposed a power iteration method to effectively recover the tensor's principal eigenvector with sparse prior. Mapping the eigenvector to the discrete solution space generates desired node-to-node correspondences. Similar to graph matching methods, hypergraph matching methods implicitly integrate geometric invariants into the objective function. But their hyperedges are able to encode more complex geometric invariants. For instance, a third-order hyperedge can provide similarity invariant using angles of the triangle defined by its three nodes; a fourth-order hyperedge can provide affine invariant by calculating the ratio between the areas of any two of the triangles defined by its four nodes. Although the hypergraph matching methods can provide geometric invariants to more complex transformation, these methods have even higher computation complexity compared with graph matching methods.

Dynamic programming also showed its effectiveness in matching tasks. Felzenszwalb and Huttenlocher [12] presented the first dynamic programming based matching method. Dynamic programming was utilized to optimize a tree model, where each node represents the best position of a pictorial part. In [16], Jiang et al. proposed a novel formulation to explicitly solve scaling and rotation parameters for all template feature points using Linearly Augmented Tree (LAT) constraints. Due to the LAT's special structure, the proposed formulation was then solved by dynamic programming. This method can handle global similarity transformations if the cost function is associative and the adjacency graph has no loops.

1.2 Our Proposed Method

Each feature matching method mentioned above supports either a specific transformation model (e.g., the similarity model in [16]) or a very small set of models (e.g., global transformation models in RANSAC methods [7], [13], [24]). In this paper, we present a unified feature matching framework that supports a large family of transformation models. We call this transformation family *the affine-function family* in which transformation models can be expressed as affine functions. This family includes many commonly used global models, such as global similarity and affine transformations, many complex transformations, such as locally-affine, Thin-plate Spline (TPS) and Free Form Deformation (FFD) models, and combinations of the above models, e.g., global similarity + local translation and so on. Some models in this family were never explored by existing methods. In this paper, we demonstrate our matching framework's performance using four transformation models in this family: 1) the global affine/similarity + local translation model, 2) two locally affine transformation models, and 3) the articulated object's transformation model.

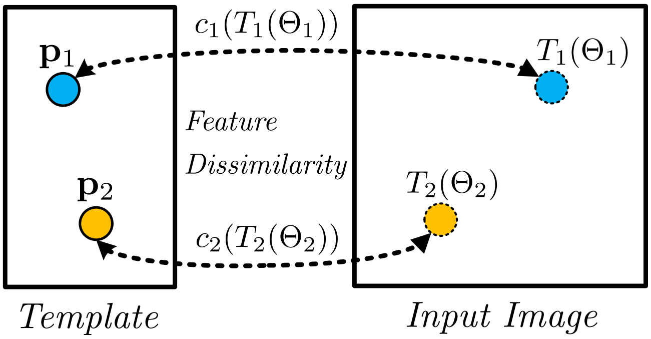


Fig. 2. Illustration of the transformation function $T_i(\cdot)$ and the feature dissimilarity function $c_i(\cdot)$ using two template feature points $\mathbf{p}_1$ and $\mathbf{p_2}$. The matching positions of $\mathbf{p}_1$ and $\mathbf{p_2}$ are calculated by their transformation functions, $T_1(\Theta_1)$ and $T_2(\Theta_2)$, respectively. Feature similarities between $\mathbf{p}_1$ and $T_1(\Theta_1)$, and between $\mathbf{p_2}$ and $T_2(\Theta_2)$ are calculated as $c_1(T_1(\Theta_1))$ and $c_2(T_2(\Theta_2))$, respectively.

Similar to other feature matching methods, feature dissimilarity values between all pairs of template points and image points are calculated before matching is performed. Our goal is to recover each template point's optimal transformation parameters such that each template point's matching position is close to an image point with similar local appearance (low dissimilarity value) while maintaining similar geometric relationships between the template points. For each template feature point, we create a convex dissimilarity function based on the dissimilarity values between it and all image points. Composition of such a convex feature dissimilarity function with a transformation model in the affine-function family has an equivalent convex optimization form. It can be efficiently performed via convex optimization techniques. We explicitly solve all template points' transformation parameters using an optimization scheme that iteratively shrinks each template point's destination region in the input image.

2 The Unified Matching Framework

2.1 Problem Formulation

Let n_t and n_s represent the numbers of feature points in the template point set and in the image point set, respectively. $\mathbf{p}_i = [x_{\mathbf{p}_i}, y_{\mathbf{p}_i}]^T, i = 1, \ldots, n_t$, and $\mathbf{q}_j = [x_{\mathbf{q}_j}, y_{\mathbf{q}_j}]^T, j = 1, \ldots, n_s$, denote the positions of the template feature points, and the positions of the image feature points, respectively. Let $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{n_t}\}$ and $\mathbf{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_{n_s}\}$ be the set of all template points' positions and the set of all image points' positions, respectively.

Unlike most feature matching methods which seek for point-to-point results, we optimize transformation parameters for each template point. Let $T_i(\Theta) : \mathbf{R}^n \to \mathbf{R}^2$ represent a transformation function that calculates the ith template feature point $\mathbf{p}_i$'s matching position in the 2D input image under certain transformation with parameters $\Theta \in \mathbf{R}^n$. The result of the $T_i(\cdot)$ function is a 2D position in the input image. We illustrate the transformation function for the affine transformation by calculating the matching position of $\mathbf{p}_i = [x_{\mathbf{p}_i}, y_{\mathbf{p}_i}]^T$ as

$$T_i^a(\Theta) = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} x_{\mathbf{p}_i} \\ y_{\mathbf{p}_i} \end{bmatrix} + \begin{bmatrix} \phi \\ \varphi \end{bmatrix}, \tag{1}$$

where $T_i^a : \mathbf{R}^6 \to \mathbf{R^2}$ calculates the matching position of the template point $\mathbf{p}_i$ under an affine transformation with parameters $\Theta = [\alpha, \beta, \gamma, \delta, \phi, \varphi]^T \in \mathbf{R}^6$. Please note that each template point $\mathbf{p}_i$'s coordinates, $[x_{\mathbf{p}_i}, y_{\mathbf{p}_i}]^T$, are constants but not variables of the transformation function $T_i(\Theta)$. Therefore, we define n_t transformation functions for n_t template points, and each template point's matching position is calculated by its corresponding $T_i(\cdot)$ function. Fig. 2 illustrates the $T_i(\cdot)$ function using two template points.

Let $T_i(\Theta) = [f_i(\Theta)\ g_i(\Theta)]^T$, where $f_i : \mathbf{R}^n \to \mathbf{R}$ and $g_i : \mathbf{R}^n \to \mathbf{R}$ are the functions calculating the x and y coordinates of $\mathbf{p}_i$'s matching position in the input image. In this paper, we focus on transformation models that can be expressed by affine functions of the transformation parameters, i.e., $f_i(\Theta)$ and $g_i(\Theta)$ are affine functions with respect to Θ. Recall that a function is affine if it is a sum of a linear function and a constant, i.e., if it has the form $f(\Theta) = a^T\Theta + c$, where $a \in \mathbf{R}^n$ and $c \in \mathbf{R}$. One such example is the above affine transformation model (1). We call the set of such transformation models the *affine-function family*.

One objective of feature matching is to match each template point $\mathbf{p}_i$ to the 2D position $T_i(\Theta)$ in the input image, such that the local appearances of the $\mathbf{p}_i$ and $T_i(\Theta)$ are similar. Therefore, a function $c_i(\mathbf{q}) : \mathbf{R}^2 \to \mathbf{R}^1$ is needed to measure the appearance (feature) dissimilarity between the template point $\mathbf{p}_i$ and a 2D position $\mathbf{q}$ in the input image. Since there are n_t template points, we need n_t feature dissimilarity functions, $c_i(\cdot), i = 1, \ldots, n_t$, one for each template point. Fig. 2 illustrates the usage of the $c_i(\cdot)$ function.

The overall objective is to find the optimal transformation parameters, $\hat{\Theta}_1, \ldots, \hat{\Theta}_{n_t}$, for template feature points, $\mathbf{p}_1, \ldots, \mathbf{p}_{n_t}$, to minimize the following objective function:

$$\begin{array}{ll} \underset{\Theta_1,\cdots,\Theta_{n_t}}{\text{minimize}} & \sum_{i=1}^{n_t} c_i(T_i(\Theta_i)) + R(\Theta_1, \Theta_2, \ldots, \Theta_{n_t}), \\ \text{subject to} & S_j(\Theta_1, \Theta_2, \ldots, \Theta_{n_t}) \leq 0, \quad j = 1, \ldots, l, \end{array} \tag{2}$$

where the term $c_i(T_i(\Theta_i))$ calculates appearance dissimilarity between the ith template point $\mathbf{p}_i$ and its matching position $T_i(\Theta_i)$ in the input image, $R(\Theta_1, \Theta_2, \ldots, \Theta_{n_t})$ is a convex term regularizing transformation parameters, and $S_j(\Theta_1, \Theta_2, \ldots, \Theta_{n_t}) \leq 0$, $j = 1, \ldots, l$ are a series of convex constraints. Both $R(\Theta_1, \Theta_2, \ldots, \Theta_{n_t})$ and $S_j(\Theta_1, \Theta_2, \ldots, \Theta_{n_t})$, $j = 1, \ldots, l$ are convex because we formulate (2) as a convex optimization problem which can be efficiently solved by convex optimization techniques. Four different implementations of the optimization model (2) are introduced in Section 3.

The main obstacle is to properly define a feature dissimilarity function $c_i : \mathbf{R}^2 \to \mathbf{R}$ such that if the transformation function $T_i(\cdot)$ is in the affine-function family, the above optimization model (2) can always be converted into an equivalent convex optimization form and therefore be efficiently optimized.

2.2 Discrete Feature Dissimilarity Function

Let $C_{i,j}$ denote the feature dissimilarity between the ith template feature point $\mathbf{p_i}$ and the jth image feature point $\mathbf{q}_j$. In this paper, we calculate $C_{i,j}$ as the L_2 distance between $\mathbf{p}_i$'s and $\mathbf{q}_j$'s feature vectors.

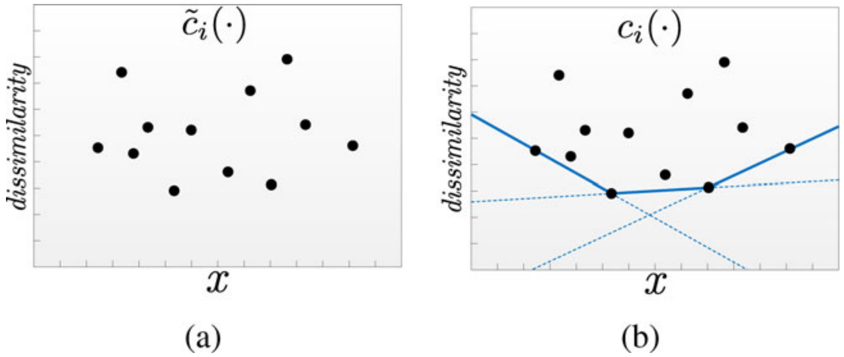


Fig. 3. Illustration of discrete and convex feature dissimilarity functions in a 1D translation + 1D feature dissimilarity space. (a) A discrete feature dissimilarity function $\tilde{c}_i(\cdot)$ for a template point $\mathbf{p}_i$. The $\tilde{c}_i(\cdot)$ function is only defined at image points' locations (shown as black dots). (b) The convex feature dissimilarity function $c_i(\cdot)$ (solid blue lines) created for that in (a). The $c_i(\cdot)$ function is the maximization of all line functions. Please note that 3D facets in (5) become 2D lines in the 2D space.

The feature similarities are all pre-calculated before the matching is performed. For each template point $\mathbf{p}_i, i = 1, \ldots, n_t$, we can define a discrete feature dissimilarity function $\tilde{c}_i : \mathbf{Q} \rightarrow \mathbf{R}$:

$$\begin{aligned}\tilde{c}_i\left(\left[x_{\mathbf{q}_1}, y_{\mathbf{q}_1}\right]^T\right) &= C_{i,1},\\ \tilde{c}_i\left(\left[x_{\mathbf{q}_2}, y_{\mathbf{q}_2}\right]^T\right) &= C_{i,2},\\ &\vdots\\ \tilde{c}_i\left(\left[x_{\mathbf{q}_{n_s}}, y_{\mathbf{q}_{n_s}}\right]^T\right) &= C_{i,n_s},\end{aligned} \tag{3}$$

where $\mathbf{Q}$ is the set containing all image feature points' positions. The domain of this function denotes that the template point $\mathbf{p}_i$ can only be matched to image feature points' positions, $\mathbf{q}_j = [x_{\mathbf{q}_j}, y_{\mathbf{q}_j}]^T, j = 1, \ldots, n_s$, with feature dissimilarity values determined by function $\tilde{c}_i(\cdot)$. This is because the calculation of $C_{i,j}$ requires feature vectors from the input image, and they are only extracted at feature points' positions. Minimization of $\tilde{c}_i(\cdot)$ results in the best matched position in the input image for the ith template point $\mathbf{p}_i$. One illustrative example of this discrete function is shown in Fig. 3a. However, the $\tilde{c}_i(\cdot)$ functions are discrete and non-convex. Optimizing (2) with the discrete feature dissimilarity functions (3) and some convex regularization terms is difficult (please see Section 3, for examples).

2.3 Convex Feature Dissimilarity Functions

To solve this problem, we relax each $\tilde{c}_i(\cdot)$ function and create a continuous and convex feature dissimilarity function $c_i(\cdot)$ which can be efficiently optimized. The above discrete functions are viewed as 3D point clouds. For the template point $\mathbf{p}_i$, all image feature points' locations, and the similarities between $\mathbf{p}_i$ and all image points can be viewed as a 3D ($n_s \times 3$) point cloud, where the third dimension represents feature dissimilarity:

$$\begin{bmatrix} x_{\mathbf{q}_1} & y_{\mathbf{q}_1} & C_{i,1}\\ x_{\mathbf{q}_2} & y_{\mathbf{q}_2} & C_{i,2}\\ \vdots & \vdots & \vdots\\ x_{\mathbf{q}_{n_s}} & y_{\mathbf{q}_{n_s}} & C_{i,n_s}\end{bmatrix}. \tag{4}$$

In Fig. 3a, we illustrate the discrete feature dissimilarity function in a 1D translation + 1D feature dissimilarity space.

We create the convex feature dissimilarity function $c_i(\cdot)$ by defining it as the lower convex hull of the ith 3D point cloud with respect to the feature dissimilarity dimension. The lower convex hull can be mathematically defined as facets in the convex hull whose normal vectors' 3rd components are less than 0 (i.e., those facets' normal vectors point downward along the dissimilarity dimension). In Fig. 3b, we illustrate the convex feature function created based on that in Fig. 3a in the 1D translation + 1D feature dissimilarity space. Let e_i denote the number of facets on $\mathbf{p}_i$'s lower convex hull, and $z_k = r_k x + s_k y + t_k$, for $k = 1, \ldots, e_i$, be the plane functions defined by these facets, where r_k, s_k, and t_k are coefficients of the kth plane function and are related to the plane's normal direction. The convex feature dissimilarity function $c_i : \mathbf{R}^2 \rightarrow \mathbf{R}$ can be defined as

$$c_i\left([x, y]^T\right) = \max_k (r_k x + s_k y + t_k), \quad k = 1, \ldots, e_i. \tag{5}$$

It denotes that the ith template point now can be matched to any location $[x, y]^T$ in the 2D image domain with a feature dissimilarity value $c_i([x, y]^T)$. Minimization of $c_i(\cdot)$ no longer matches the ith template point to only image point locations but any location in the 2D input image. Although optimizing (5) looks difficult, it is equivalent to a linear program in essence:

$$\begin{aligned}\underset{x,y}{\text{minimize}} \;\; c_i\left([x, y]^T\right) \;\Leftrightarrow\; \underset{u_i,x,y}{\text{minimize}} \;\; & u_i\\ \text{subject to} \;\; & r_k x + s_k y + t_k \leq u_i,\\ & k = 1, \ldots, e_i,\end{aligned} \tag{6}$$

where u_i is an auxiliary variable representing the upper bound of $c_i([x, y]^T)$. By transforming the optimization of $c_i([x, y]^T)$ into the equivalent linear program, it can be efficiently optimized.

The lower convex hull technique has an intuitive interpretation (Fig. 3). The lower convex hulls are lower bounds of the discrete feature dissimilarity functions. They are more likely to include image points with lower similarities compared with their neighbors. However, when features are not distinctive, this technique may not generate satisfactory lower bounds. We will introduce an iterative technique to empirically and gradually obtain tighter lower bounds in Section 4.

2.4 Composition with Transformation Models in the Affine-Function Family

Instead of directly searching for point-to-point correspondences, we prefer calculating template points' matching positions using some geometric transformation models and determining their feature dissimilarities using (5).

Recall that $T_i(\Theta) : \mathbf{R}^n \rightarrow \mathbf{R}^2$ calculates the template point $\mathbf{p}_i$'s matching position using an affine-function model with parameters $\Theta \in \mathbf{R}^n$. $c_i(T_i(\Theta_i))$ is the composition of the convex feature dissimilarity function $c_i(\cdot)$ with the template point's transformation function, $T_i(\Theta_i)$, with $\mathbf{p}_i$'s parameters Θ_i. It measures the feature dissimilarity between the template point $\mathbf{p}_i$ and its matching position $T_i(\Theta_i)$ in the input image. One important property of the $c_i(T_i(\Theta_i))$ is that its minimization can always be reformulated into an equivalent convex optimization model with respect to the transformation parameters Θ_i.

It is easy to show this property. Recall $T_i(\Theta_i) = [f_i(\Theta_i)\ g_i(\Theta_i)]^T$. Substituting x and y in (6) with $f_i(\Theta_i)$ and $g_i(\Theta_i)$ leads to the following optimization model equivalent to minimizing $c_i(T_i(\Theta_i))$ with respect to Θ_i:

$$\begin{aligned} \underset{\Theta_i}{\text{minimize}}\, c_i(T_i(\Theta_i)) \Leftrightarrow \underset{u_i,\Theta_i}{\text{minimize}} \quad & u_i \\ \textit{subject to} \quad & r_k f_i(\Theta_i) + s_k g_i(\Theta_i) \\ & \quad + t_k - u_i \leq 0, \\ & k = 1, \ldots, e_i. \end{aligned} \tag{7}$$

Since both $f_i(\Theta)$ and $g_i(\Theta)$ are affine functions of Θ_i, $r_k f_i(\Theta_i) + s_k g_i(\Theta_i) + t_k - u_i$ for $k = 1, \ldots, e_i$ are also affine functions of u_i and Θ_i. Therefore, (7) is a convex optimization problem.

It indicates that $c_i(T_i(\Theta_i))$ can be efficiently optimized via convex optimization techniques. We show how the composition is done with the affine transformation (1): substituting $f_i(\Theta) = \alpha x_{\mathbf{p}_i} + \beta y_{\mathbf{p}_i} + \phi$ and $g_i(\Theta) = \gamma x_{\mathbf{p}_i} + \delta y_{\mathbf{p}_i} + \varphi$ into (7) leads to an actual optimization model using global affine transformation models. In the remaining part of this paper, we write minimization of $c_i(T_i(\Theta_i))$ directly to simplify the notation. Please keep in mind it can be efficiently optimized as (7).

2.5 The Overall Objective Function

Since $c_i(T_i(\Theta_i))$ can be reformulated into an equivalent convex form, (2) can be converted into a convex form. The optimal transformation parameters for all template points, $\hat{\Theta}_1, \ldots, \hat{\Theta}_{n_t}$, can then be obtained by minimizing the model (2).

One advantage of the above objective function is that it no longer requires template points being matched to only image points' positions. Instead, it can be matched to any position in the 2D input image. This property is very useful when some template points' corresponding feature points are not detected in the input image. For those template points, their transformation parameters can be inferred by the parameters of their neighbors. However, if template points are only allowed to be match to image points' positions, our method still can obtain such a result with a subsequent step (see Section 4).

2.6 Related Work

Our work is most related to the linear programming based matching methods [14], [15], [19], [32]. These methods look for a matching function $m : \mathbf{P} \rightarrow \mathbf{Q}$ that maps every template point $\mathbf{p}_i \in \mathbf{P}$ to an image feature point $\mathbf{q}_j \in \mathbf{Q}$. The matching function $m(\cdot)$ is modeled as a series of linear variables. Jiang et al. [14] formulated the matching problem as

$$\underset{m}{\text{minimize}} \sum_{i=1}^{n_t} \left(\tilde{c}_i(m(\mathbf{p}_i)) + \lambda \sum_{j \in \mathcal{N}_{\mathbf{p}_i}} R\big(\mathbf{p}_i, \mathbf{p}_j; m(\mathbf{p}_i), m(\mathbf{p}_j)\big) \right), \tag{8}$$

where $\tilde{c}_i(\cdot)$ is the discrete feature dissimilarity function defined in (3), R is a geometric regularization term that measures the geometric similarity between the vector $\mathbf{p}_j \rightarrow \mathbf{p}_i$ and the vector $m(\mathbf{p}_j) \rightarrow m(\mathbf{p}_i)$, and $\mathbf{p}_j \in \mathcal{N}_{\mathbf{p}_i}$ is a pre-defined neighbor of $\mathbf{p}_i$. The regularization term is defined based on 2D vectors specified by pairs of neighboring template points. Such 2D vectors are only invariant to simultaneous translations of their pairs of feature points. Jiang and Yu [15] followed this linear programming framework and utilized the same 2D vectors. Global rotation and scaling are solved for all 2D vectors simultaneously to achieve global similarity invariant. Li et al. [19] represented each template point $\mathbf{p}_i$ as an affine combination of its neighbors $\mathcal{N}_{\mathbf{p}_i}$, i.e., $\|\mathbf{p}_i - \sum_{\mathbf{p}_j \in \mathcal{N}_{\mathbf{p}_i}} w_{ij}\, \mathbf{p}_j\|_1 = 0, \forall \mathbf{p}_i \in \mathbf{P}$, where $\sum_{\mathbf{p}_j \in \mathcal{N}_{\mathbf{p}_i}} w_{ij} = 1$. Locally-affine invariants can be obtained by reconstructing matched image points using the same affine combination coefficients w_{ij}. The reconstruction errors were used to define the geometric regularization term:

$$\underset{m}{\text{minimize}} \sum_{i=1}^{n_t} \left(\tilde{c}_i(m(\mathbf{p}_i)) + \lambda \left\| m(\mathbf{p}_i) - \sum_{\mathbf{p}_j \in \mathcal{N}_{\mathbf{p}_i}} w_{ij}\, m(\mathbf{p}_j) \right\|_1 \right). \tag{9}$$

In our proposed method, instead of seeking for a matching function that specifies point-to-point correspondences, we model all template points' matching positions by $T_i(\cdot)$ functions and solve their transformation parameters. The previous methods [14], [15], [19] used lower convex hulls to choose candidate matching points from the image set for each template point. The number of variables in the linear program can be significantly reduced in this way. Although inspired by them, our method uses the lower convex hull in an essentially different way. We treat it as relaxation of the discrete dissimilarity function. More importantly, because we optimize each template point's transformation parameters, we compose it with affine-function transformations to support a large family of transformations. We also showed that the composition can always be converted into an equivalent convex optimization form.

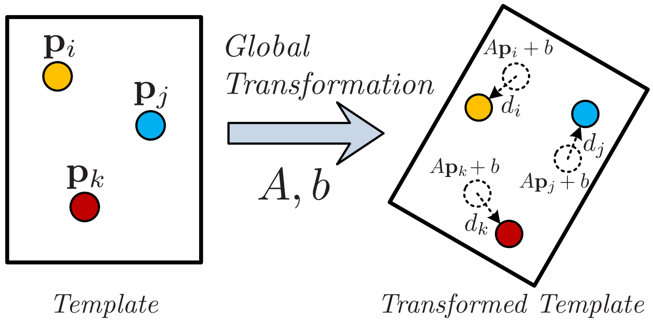


Fig. 4. Illustration of the global affine/similarity + local translation model. After a global transformation with parameters A and b, the three matched positions individually translate for d_i, d_j, and d_k to better fit the input image.

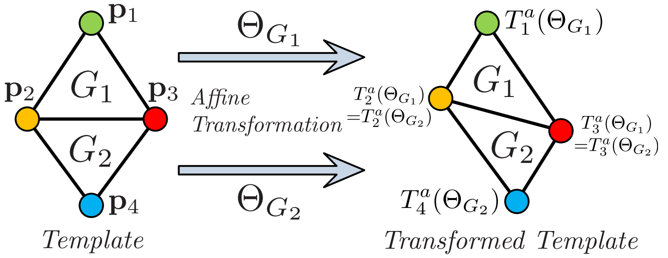


Fig. 5. Illustration of the locally affine transformation model I using four points ($\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$) in two triangles ($G_1$ and G_2). To maintain the mesh topology after matching, the equality constraints (14) on $\mathbf{p}_2$ and $\mathbf{p}_3$ should be satisfied as $T_2^a(\Theta_{G_1}) = T_2^a(\Theta_{G_2})$, and $T_3^a(\Theta_{G_1}) = T_3^a(\Theta_{G_2})$.

3 Four Transformation Models in the Affine-Function Family

In this section, we present four transformation models in the affine-function family to show the flexibility of our proposed framework: 1) the global affine/similarity + local translation model, 2) two locally affine transformation models, and 3) the articulated object's transformation model. For different models, different geometric regularization terms and convex constraints are used in (2). Note that transformation models are not limited to the ones we mention here. Any model in the affine-function family can be used in our framework.

3.1 The Global Affine/Similairty + Local Translation Model

Our global transformation model assumes a global affine or similarity transformation and additional small local deformations between template points and their matched positions. Therefore, all template points should share a common set of global transformation parameters. To model local deformations and to better fit the input image locally, each matched point is allowed to translate individually for a small distance. We make small changes to $T_i^a(\Theta)$ function in (1) and calculate the matching position for template point $\mathbf{p}_i$ as

$$T_i^g(\Theta_i) = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} x_{\mathbf{p}_i} \\ y_{\mathbf{p}_i} \end{bmatrix} + \begin{bmatrix} \phi \\ \varphi \end{bmatrix} + \begin{bmatrix} \phi_i \\ \varphi_i \end{bmatrix} \tag{10}$$

$$= A\mathbf{p}_i + b + d_i, \tag{11}$$

where A represents the 2×2 global transformaton matrix, b is the 2×1 global translation vector, and $d_i = [\phi_i, \varphi_i]^T$ is $\mathbf{p}_i$'s local translation vector. The template point $\mathbf{p}_i$'s transformation parameters Θ_i therefore consist of common global transformation parameters, A and b, and an individual local translation vector d_i (Fig. 4). To penalize local deformations that are too large, the square of the distance by which each matched point translates locally, $\|d_i\|_2^2$, is regularized. The above transformation model and regularization terms result in the following optimization model,

$$\underset{A, b, d_1, \ldots, d_{n_t}}{\text{minimize}} \quad \sum_{i=1}^{n_t} \Big(c_i(A\mathbf{p}_i + b + d_i) + w_g \|d_i\|_2^2 \Big), \tag{12}$$

$$\text{subject to} \quad \alpha = \delta,\ \beta = -\gamma,$$
$$\text{(when the global transformation is similarity transformation)} \tag{13}$$

where w_g is the parameter that weighs the feature dissimilarity terms and the local translation regularization terms, and $\alpha, \beta, \gamma, \delta$ are the four elements of the A matrix as defined in (10). The constraint (13) is applied when the transformation is global similarity instead of affine. The above optimization model provides more robust matching results when template points are known to undergo mostly an affine or similarity transformation.

3.2 Locally Affine Transformation Model I

If the template points undergo complex transformations that cannot be described by the global affine or similarity transformation model, we propose to approximate the template points' transformation with a locally affine transformation model.

The first locally affine model achieves locally affine invariance by assuming an affine transformation for every three neighboring template points. We first use Delaunay Triangulation to obtain a triangulated mesh from the 2D template points. Then every three points defining a triangle on the mesh share a same affine transformation, in other words, every three template points in a triangle share a common set of affine transformation parameters (Fig. 5). Let m denote the number of triangles in the triangulated mesh, $G_1, G_2, \ldots, G_m$ denote the m sets consisting of the points in the 1st, 2nd, …, mth triangles, and $\Theta_{G_v} \in \mathbf{R}^6$ denote the affine transformation parameters for template points in the vth triangle. If the vth and the wth triangles share a common edge, we call them two neighboring triangles and denote them as $v \in \mathcal{N}_w, w \in \mathcal{N}_v$.

One template point might be in several triangles. To make sure one template point's matching positions calculated by different triangles' transformation parameters are the same position, the following equality constraints need to be added into the optimization model:

$$T_i^a(\Theta_{G_v}) = T_i^a(\Theta_{G_w}), \quad \begin{array}{l} \text{for all } i = 1, \ldots, n_t, \\ \text{for all } v,\ w = 1, \ldots, m, \end{array} \tag{14}$$

where $\mathbf{p}_i \in G_v$ and $\mathbf{p}_i \in G_w$.

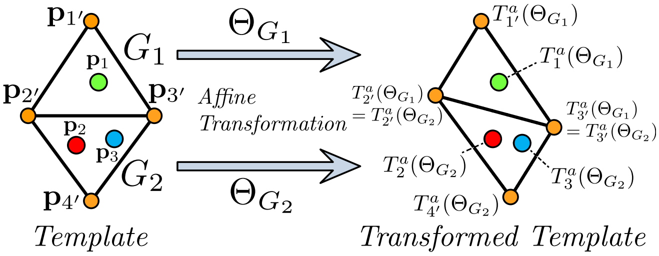


Fig. 6. Illustration of the locally affine transformation model II using three template feature points ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$), and four mesh points ($\mathbf{p}_{1'}$, $\mathbf{p}_{2'}$, $\mathbf{p}_{3'}$, $\mathbf{p}_{4'}$) in two triangles (G_1 and G_2). To maintain the mesh topology after matching, the equality constraints on $\mathbf{p}_{2'}$ and $\mathbf{p}_{3'}$ should be satisfied as $T^a_{2'}(\Theta_{G_1}) = T^a_{2'}(\Theta_{G_2})$, and $T^a_{3'}(\Theta_{G_1}) = T^a_{3'}(\Theta_{G_2})$.

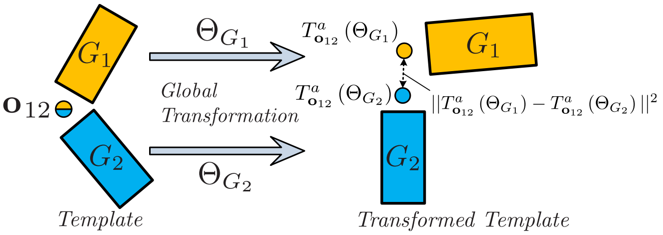

G_1
$\mathbf{o}\,12$
G_2
Θ_{G_1}
Global Transformation
Θ_{G_2}
$T^a_{\mathbf{o}_{12}}(\Theta_{G_1})$
$T^a_{\mathbf{o}_{12}}(\Theta_{G_2})$
$||T^a_{\mathbf{o}_{12}}(\Theta_{G_1}) - T^a_{\mathbf{o}_{12}}(\Theta_{G_2})||^2$
G_1
G_2
Template
Transformed Template

Fig. 7. Illustration of the articulated object's transformation model with a 2-part object. $\mathbf{o}_{12}$ is the junction point between parts G_1 and G_2. The squared distance between the transformed junction points $T^a_{\mathbf{o}_{12}}(\Theta_{G_1})$ and $T^a_{\mathbf{o}_{12}}(\Theta_{G_2})$ is used for regularization.

The above equality constraints are illustrated in Fig. 5 with four points, $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$, in two triangles, $G_1 = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ and $G_2 = \{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$. The equality constraints should be applied to $\mathbf{p}_2$ and $\mathbf{p}_3$ as $T^a_2(\Theta_{G_1}) = T^a_2(\Theta_{G_2})$, $T^a_3(\Theta_{G_1}) = T^a_3(\Theta_{G_2})$.

The mesh formed by the matched points should maintain its smoothness. The differences between neighboring triangles' transformation parameters are penalized as regularization terms in the optimization model. The final objective function for the locally affine model can be expressed as

$$\begin{aligned} &\underset{\Theta_{G_1},\dots,\Theta_{G_m}}{\text{minimize}} \left(\sum_{v=1}^{m} \sum_{\mathbf{p}_i \in G_v} c_i(T^a_i(\Theta_{G_v})) + w_l \sum_{v=1}^{m} \sum_{u \in \mathcal{N}_v} \|\Theta_{G_v} - \Theta_{G_w}\|_1 \right), \\ &\text{subject to} \quad \textit{The Equality Constraints in } (14), \end{aligned} \tag{15}$$

where w_l is the parameter that weighs the feature dissimilarity terms and the mesh smoothness regularization terms. The above model can approximate very complex transformations.

3.3 Locally Affine Transformation Model II

In locally affine model I, we create a mesh by computing a Delaunay triangulation on the template feature points. In some matching tasks, instead of having feature points as nodes in the mesh, a triangulated mesh that better represents the movements of the template points is given in advance (see cases in Section 5.5). To take advantage of the pre-given mesh, we present another locally affine model which transforms a given triangulated mesh with feature points lying within its triangles. We let all feature points in a same triangle share a common set of affine transformation parameters (Fig. 6). We reuse the notation in Section 3.2 without causing confusion. In addition, let $\mathbf{p}_{i'}$ represent the ith node point on the pre-given mesh, and $T^a_{i'}(\Theta)$ calculates its matching position using affine transformation with parameters Θ.

The locally affine model II is illustrated in Fig 6. Equality constraints similar to (14) are now applied to mesh points that are in several triangles simultaneously. The final optimization model for locally affine model II is the same as (19) except that the equality constraints are now applied to mesh points $\mathbf{p}_{i'}$.

3.4 Articulated Object's Transformation Model

There are many types of objects, e.g., human bodies, have several connected parts that undergo rigid transformations relative to their neighbors in the object's kinematic tree. Let m denote the number of rigid parts in an object, $G_1, G_2, \dots, G_m$ denote the m sets consisting of the points in the 1st, 2nd, …, mth parts, and $\Theta_{G_v} \in \mathbf{R}^6$ denote the parameters for the transformation of template points in the vth part.

For every pair of connected parts v and w, we model a junction point $\mathbf{o}_{vw}$ denoting the connecting point between the two parts. During the matching process, it is deformed according to both parts' transformation parameters to $T^a_{\mathbf{o}_{vw}}(\Theta_{G_v})$ and $T^a_{\mathbf{o}_{vw}}(\Theta_{G_w})$, where $T^a_{\mathbf{o}_{vw}}(\Theta)$ is the matched position of $\mathbf{o}_{vw}$ calculated by an affine transformation with parameters Θ. To maintain connectivity of the two parts v and w, the squared distance between the two matched junction points, $\|T^a_{\mathbf{o}_{vw}}(\Theta_{G_v}) - T^a_{\mathbf{o}_{vw}}(\Theta_{G_w})\|_2^2$, should be minimized and is used as a regularization term. The transformation model for a two-part articulated object is illustrated in Fig. 7. The final optimization model for the transformation model of a general articulated object is then defined as

$$\begin{aligned} &\underset{\Theta_{G_1},\cdots,\Theta_{G_m}}{\text{minimize}} \left(\sum_{v=1}^{m} \sum_{\mathbf{p}_i \in G_v} c_i(T^a_i(\Theta_{G_v})) + w_a \sum_{\mathbf{o}_{vw}} \left\|T^a_{\mathbf{o}_{vw}}(\Theta_{G_v}) - T^a_{\mathbf{o}_{vw}}(\Theta_{G_w})\right\|_2^2 \right), \\ &\text{subject to} \quad \alpha_{G_v} = \delta_{G_v},\ \beta_{G_v} = -\gamma_{G_v}, \\ &\qquad\qquad\quad (\text{if the } v\text{th part undergoes similarity transformation}) \end{aligned} \tag{16}$$

where w_a is the parameter that weighs the feature dissimilarity terms and the part connectivity regularization terms, A_{G_v} is the vth part's 2×2 transformation matrix defined similarly to that in (11), and α_{G_v}, β_{G_v}, γ_{G_v}, δ_{G_v} are its four elements.

4 Numerical Scheme and Analysis

The convex feature dissimilarity functions $c_i(\cdot)$ are created by relaxing the discrete functions $c_i(\cdot)$. When features are distinctive, i.e., the feature similarities of most template points to their corresponding image point locations are significantly lower than those to other image points, the lower convex hull relaxation provides satisfactory lower bounds to the discrete functions $c_i(\cdot)$. However, when features are not distinctive, the relaxation may generate functions with "broad" low-cost regions. To solve this problem, we use a technique similar to that in [14]. We iteratively create these convex feature functions using fewer and fewer image feature points and gradually obtain tighter lower bounds.

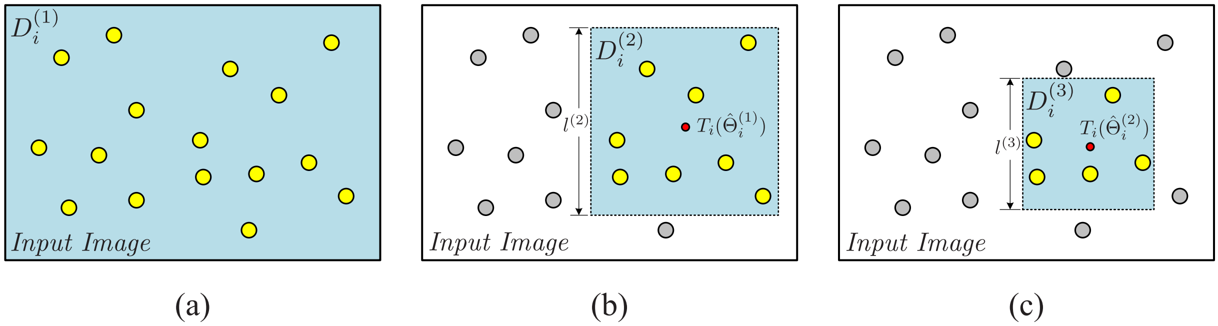


Fig. 8. (a) In the first iteration, for template point $\mathbf{p}_i$, all image points (yellow dots) in the input image are used to create the $c_i(\cdot)$ function. (b) In the second iteration, trust region $D_i^{(2)}$ with side length $l^{(2)}$ is centered at previous iteration's result $T_i(\hat{\Theta}_i^{(1)})$ (the red dot). Only image points in $D_i^{(2)}$ (yellow dots) are used to create the $c_i(\cdot)$ function. (c) Similar operations as those in the second iteration (b) are performed in latter iterations.

In the first iteration, for each template point, we use all image feature points' positions and feature similarities as in (4) to create its convex feature dissimilarity function $c_i(\cdot)$. Those convex functions are then used in our proposed optimization model (2), and template points are matched to positions calculated with the optimal transformation parameters $\hat{\Theta}_1^{(1)}, \ldots, \hat{\Theta}_{n_t}^{(1)}$, where $\hat{\Theta}_i^{(k)}$ denotes the optimized transformation parameters for $\mathbf{p}_i$ in the kth iteration. In Fig. 8a, yellow dots represent that all feature points in the input image are chosen to create the $c_i(\cdot)$ function for $\mathbf{p}_i$.

In the second iteration, for each template point $\mathbf{p}_i$, we set a "trust region" centered at the first iteration's optimal matching position $T_i(\hat{\Theta}_i^{(1)}) = [f_i(\hat{\Theta}_i^{(1)})\ g_i(\hat{\Theta}_i^{(1)})]^T$ in the 2D input image. Mathematically, $\mathbf{p}_i$'s trust region in the second iteration is defined as

$$\begin{aligned} D_i^{(2)} = \Big\{ [x,\, y]^T \in \mathbf{R}^2 \big| f_i\big(\hat{\Theta}_i^{(1)}\big) - l^{(2)}/2 \leq x \leq f_i\big(\hat{\Theta}_i^{(1)}\big) \\ + l^{(2)}/2, g_i\big(\hat{\Theta}_i^{(1)}\big) - l^{(2)}/2 \leq y \leq g_i\big(\hat{\Theta}_i^{(1)}\big) + l^{(2)}/2 \Big\}, \end{aligned} \tag{17}$$

where $l^{(2)}$ is the side length of the trust regions in the second iteration. Each trust region is smaller than the entire image and therefore includes fewer image points. We can then generate convex feature dissimilarity functions $c_i(\cdot)$ using only those image points inside the trust regions and obtain tighter lower bounds for the discrete $\tilde{c}_i(\cdot)$ functions. In Fig. 8b, the shaded area illustrates the template point $\mathbf{p}_i$'s trust region centered at $T_i(\hat{\Theta}_i^{(1)})$ (the red dot), where only image points (yellow dots) inside it are used to create the convex feature dissimilarity function. Similar operations are performed in the subsequent iterations, where feature dissimilarity functions are created based on smaller and smaller trust regions centered at the previous iterations' results (Fig. 8c).

In each iteration, each template point's candidate matching positions are divided into two areas, its trust region and the trust region's complement. There are 2^{n_t} possible combinations in total but only one combination was tested, where n_t is the number of template feature points. Although a globally optimal solution is not guaranteed in this way because the other $2^{n_t} - 1$ combinations in each iteration are not tested, this scheme gains much more in lowering computation complexity while keeping empirically high matching accuracy. The running speed of our method depends mainly on the number of template points and depends much less on the number of image points. For a matching case with a 100-point template, our method usually needs no more than eight iterations and each iteration takes no more than $2\ s$ using a MATLAB implementation with CVX on a computer with a $3.0\ GHz$ Core 2 Duo CPU.

The gap between the optimal solution and our method's solution depends on the distinctiveness of the feature points. When the feature points are very distinctive, the relaxation of discrete dissimilarity function using the lower convex hull is tight, whereas the relaxation could bring large errors if template feature points are very indistinctive. Although we cannot provide a quantitative bound on the gap here, we observed in our experiments (see the third experiment in Section 5.1) that the chance of getting large errors due to the relaxation is very small.

Although our method calculates each template point's optimal transformation parameters, if a point-to-point result is desired, we propose to exhaustively search for the best matched image point, $\hat{\mathbf{q}}_{\mathbf{p}_i}$, for each template point $\mathbf{p}_i$ after its optimal transformation parameters $\hat{\Theta}_i$ is obtained:

$$\hat{\mathbf{q}}_{\mathbf{p}_i} = \arg \min_{\mathbf{q}_j} (\|\mathbf{q}_j - T_i(\hat{\Theta}_i)\|_2 + w_h \tilde{c}_i(\mathbf{q}_j)). \tag{18}$$

The above function denotes that the best matched image point $\hat{\mathbf{q}}_{\mathbf{p}_i}$ for $\mathbf{p}_i$ should be close to the optimal matching result obtained from (2) as well as having a small feature dissimilarity value with $\mathbf{p}_i$. w_h is the parameter that weighs the feature dissimilarity and the distance to the optimal result obtained from (2).

5 Experiments

In this section, we present experiments to test the performance of our method. Except for the experiments in Sections 5.1 and 5.2, where Shape Context [3] with its default parameter settings is used as features, SIFT [22] feature points are used for all other experiments. For each experiment, we tested three weight values, 0.1, 1 and 5, as the weight factor between the feature dissimilarity terms and regularization terms; the weight resulting in the best matching result was chosen. In all our experiments, we empirically and gradually shrink each template point's trust region from the image size to a 15×15 square.

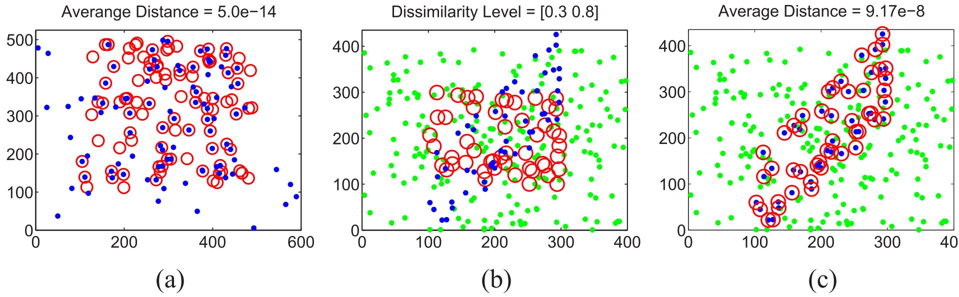


Fig. 9. (a) An example point matching case using Shape Context [4] as features with $50\% \times n_t$ deleted template points and $50\% \times n_t$ outlier points. Red circles and blue dots represent template points and image points, respectively. (b)-(c) An example point matching case using randomly set feature dissimilarity values described in the second paragraph of Section 5.1. Red circles represent template feature points. Blue and green dots represent corresponding and non-corresponding image feature points, respectively. (b) The matching case before matching. (c) The matching case after matching.

5.1 Synthetic Data

To compare with the method in [15], we slightly changed the experimental setup in [15] and created random-point matching cases with Shape Context [3] features. The global similarity model was used in this comparison. To generate each matching case, we first used points randomly spread in the region $[100, 500] \times [100, 500]$ as template feature points. Then, to generate image points from the template points, we randomly deleted $h\% \times n_t$ points from the image point set to simulate the effect of failing to detect some feature points, and added $h\% \times n_t$ randomly spread points in $[0, 600] \times [0, 600]$ as outliers. The average L_2 distance between the transformed template points and their known corresponding image points is calculated as the matching error for each matching case. One matching case with 50 percent outliers and 50 percent mis-detection using our method is shown in Fig. 9a. For each 10, 20, 30, 40 and 50 percent outlier and mis-detection level, we created 100 matching cases and matched them using our global similarity + local translation model (12) and the method in [15]. The statistics of errors on the matching cases using the two methods are compared in Table 1. Our model (12) provides more constraints for this task and therefore is able to tolerate more outliers and missing points.

To compare with the method in [19], we randomly set feature dissimilarity values between all pairs of template and image points. The global affine model was used in this comparison. For each matching case, we randomly generated 50 points in $[100, 300] \times [100, 300]$ as the template point set. To generate an image point set from the template point set, we first transformed the template point set under affine transformations with shear parallel to the x dimension. We tested three shear values: 0.5, 1.0 and 1.5. The transformed template points with 250 additional random points in $[0, 400] \times [0, 400]$ were then used as the image point set. One example of such a matching case with 1.0 shear is shown in Figs. 9b and 9c. Since we know the correspondences between template points and image points, we set the feature similarities between non-corresponding points to lie randomly in the range of $[0.5, 1.0]$, and set feature similarities between corresponding points randomly in the ranges of $[0.2, 0.7]$, $[0.3, 0.8]$, or $[0.4, 0.9]$ to simulate three levels of feature distinctiveness. For each shear value and dissimilarity level combination, we generated 100 random matching cases as described above. L_2 distances between known corresponding points after matching are calculated as errors. As shown in Table 2, our method generates smaller errors than the LP method in [19]. However, both methods have larger chances to fail when the corresponding points' feature dissimilarities are in the range of $[0.4, 0.9]$.

To determine the robustness of our method against occlusion, we combined the setup of the previous two experiments. For each matching case, we randomly generated 50 points in $[100, 300] \times [100, 300]$ as template points. To generate an image point set, we first sorted the template point set according to their x coordinates in the descending order. The first $h\% \times n_t$ points, which are the right portion of the point set, are deleted after sorting. We then transformed the remaining points under affine transformations with 1.0 shear parallel to the x dimension. Finally, transformed points with 250 additional random points in

TABLE 1
Means and Standard Deviations of Errors for Matching Synthetic Data Using Our Proposed Method and the Method in [15] with Shape Context Features

	Errors by [15]	Errors by Our Method
$h\% = 10\%$	1.34 ± 4.06	$\mathbf{0.00 \pm 0.00}$
$h\% = 20\%$	4.37 ± 11.29	$\mathbf{0.10 \pm 1.08}$
$h\% = 30\%$	6.59 ± 13.01	$\mathbf{0.53 \pm 2.03}$
$h\% = 40\%$	10.23 ± 13.88	$\mathbf{3.27 \pm 6.23}$
$h\% = 50\%$	19.94 ± 21.66	$\mathbf{18.72 \pm 17.81}$

TABLE 2
Means and Standard Deviations of Errors for Matching Synthetic Data Using Our Proposed Method and the Method in [19] with Randomly Set Feature Dissimilarity Values

Dissimilarities	Shear	Errors by [19]	Errors by Our Method
$[0.2, 0.7]$	0.5	0.00 ± 0.00	0.00 ± 0.00
	1.0	0.00 ± 0.00	0.00 ± 0.00
	1.5	0.40 ± 3.29	$\mathbf{0.00 \pm 0.00}$
$[0.3, 0.8]$	0.5	0.34 ± 2.30	$\mathbf{0.01 \pm 0.09}$
	1.0	2.36 ± 8.20	$\mathbf{0.01 \pm 0.03}$
	1.5	12.42 ± 23.61	$\mathbf{1.35 \pm 6.72}$
$[0.4, 0.9]$	0.5	20.21 ± 27.38	$\mathbf{8.03 \pm 16.92}$
	1.0	37.14 ± 35.17	$\mathbf{19.80 \pm 29.54}$
	1.5	62.10 ± 46.92	$\mathbf{37.40 \pm 36.66}$

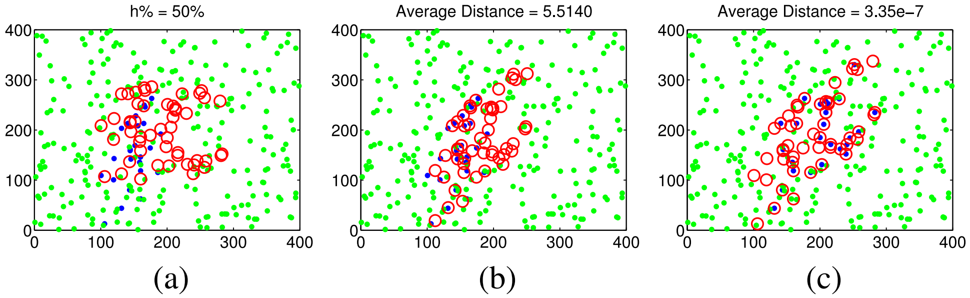


Fig. 10. An example point matching case with aggregated missing corresponding points (a) before matching, (b) after matching, and (c) its baseline matching case's result. Red circles represent template feature points. Blue and green dots represent corresponding and non-corresponding image feature points, respectively.

TABLE 3
Means and Standard Deviations of Errors for Matching Synthetic Data to Determine the Robustness of the Proposed Method against Occlusion

$h\%$	10%	20%	30%	40%	50%
Errors of Simulated Occlusion Cases	0.54 ± 2.46	1.19 ± 3.56	4.30 ± 0.67	7.44 ± 11.29	10.74 ± 14.99
Errors of Simulated Mis-detection Cases	$\mathbf{0.18 \pm 0.94}$	$\mathbf{0.37} \pm \mathbf{1.45}$	$\mathbf{1.75 \pm 0.59}$	$\mathbf{4.49 \pm 11.61}$	$\mathbf{6.95 \pm 14.51}$

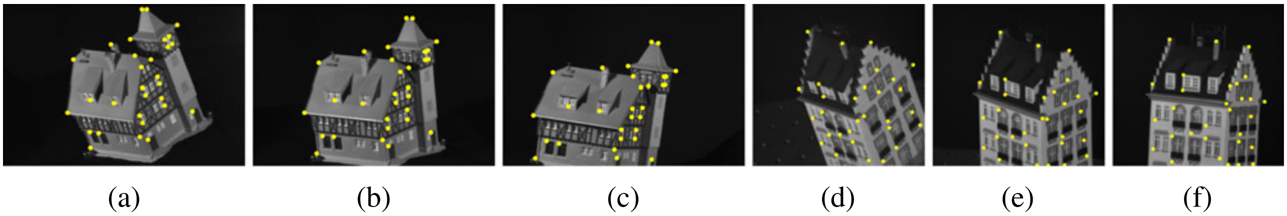


Fig. 11. Example frames with manual labeling from the CMU *house* and *hotel* sequences. (a) The first, (b) the 56th, and (c) the 111st frames in the *house* sequence. (d) The first, (e) the 51st, and (f) the 101st frames in the *hotel* sequence.

$[0, 400] \times [0, 400]$ were used as the image point set. Similar to the previous experiment, we set the feature similarities between non-corresponding points to lie randomly in the range of $[0.5, 1.0]$, and set feature similarities between corresponding points randomly in the range of $[0.3, 0.8]$. We also created baseline matching cases by randomly deleting template points in the first step instead of deleting sorted template points. Randomly deleting points can simulate the effect that some template points' corresponding ones are not detected in the input image. We then used the global affine model to solve both matching cases. For $h\% = 10\%$, 20%, 30%, 40% and 50%, we create 100 matching cases at each level. L_2 distances between known corresponding points after matching are calculated as errors. Fig. 10 shows one matching case and its baseline matching case. As shown in Table 3, our method would generate greater errors if the "missing" points aggregate. With the same number of missing points, our method is more vulnerable to occlusion than randomly mis-detected points.

5.2 CMU House and Hotel Sequences [2], [1]

We followed the experimental setup in [6] and tested our method's performance on the CMU *House* and *Hotel* sequences. Shape Context [3] is used as features. The two sequences consist of 111 and 101 frames, respectively. Each frame is manually labeled with the same 30 landmarks.[1]

Example frames of the two sequences with their manual labeling are shown in Fig. 11. We created matching cases by taking two frames in a same sequence but separated by a specific number of in-between frames. The separation number was increased from 10 to 110 for the *house* sequence and from 10 to 100 for the *hotel* sequence. Note that the highest frame separation level contains only 1 image pair for each sequence.

The two image sets are created by viewing objects from different view points. We chose the locally affine model I to handle the two image sets. However, in this experiment, the prior knowledge that each template point must be matched to one and only one image point can be used to better constrain the matching result. Additional variables and constraints are therefore needed. We first define an additional binary variable matrix $X \in \{0,1\}^{30\times30}$, whose entry at its ith row and jth column, $X_{i,j}$, represents the matching result between the template point $\mathbf{p}_i$ and the image point $\mathbf{q}_j$ being either "Yes" (1) or "No" (0). The constraint, $\forall \mathbf{p}_i \in G_v : T_i^a(\Theta_{G_v}) = \sum_{j=1}^{30}(X_{i,j}\mathbf{q}_j)$, for all G_1, $G_2, \ldots, G_m$, denotes that each template point, $\mathbf{p}_i$, must be matched to the position of the image point specified by the binary matrix X as $\sum_{j=1}^{30} X_{i,j}\mathbf{q}_j$. Let $\mathbf{1}_{30}$ represents a vector of 30 "1"s. To make sure each template point being matched to one and only one image point, the two constraints $X\mathbf{1}_{30} = \mathbf{1}_{30}$ and $X^T\mathbf{1}_{30} = \mathbf{1}_{30}$ are also added. Please note that similar one-to-one constraints are also used in [19]. With the additional variables and constraints, the

[1] The manual labeling can be obtained from `http://tiberiocaetano.com/data/`.

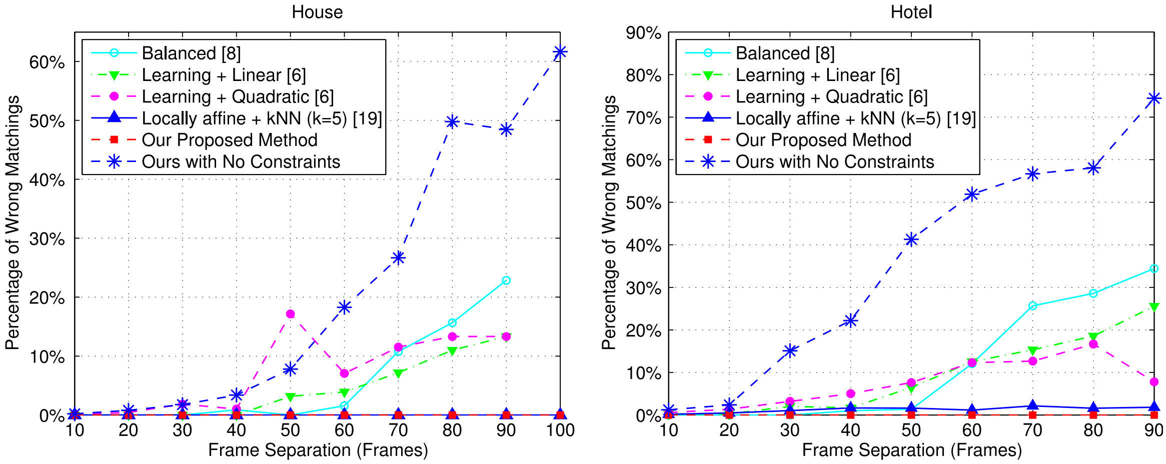


Fig. 12. Matching errors by different methods with varying frame separation levels on (left) the CMU *house* sequence and (right) the CMU *hotel* sequence. The results by methods in [6], [8] are obtained from [6]. Note that the highest frame separation level contains only one image pair for each sequence. Our method with locally affine model I generates 0.0 percent matching errors for both the *house* and the *hotel* sequences and outperforms all other methods.

locally affine model I is modified to

$$
\begin{aligned}
&\underset{\Theta_{G_1},\cdots,\Theta_{G_m},X}{\text{minimize}} \left(\sum_{v=1}^{m} \sum_{\mathbf{p}_i \in G_v} c_i(T_i^a(\Theta_{G_v})) + w_l \sum_{v=1}^{m} \sum_{u \in \mathcal{N}_v} \|\Theta_{G_v} - \Theta_{G_w}\|_1 \right),\\
&\text{subject to} \quad \textit{The Equality Constraints in } (14),\\
&\qquad X \in [0,1]^{30\times 30}, X\mathbf{1}_{30} = \mathbf{1}_{30},\ X^T\mathbf{1}_{30} = \mathbf{1}_{30},\\
&\qquad \forall \mathbf{p}_i \in G_v : T_i^a(\Theta_{G_v}) = \sum_{j=1}^{30} (X_{i,j}\mathbf{q}_j), \text{for all } G_1, G_2, \ldots, G_m,
\end{aligned}
\tag{19}
$$

where the binary constraints, $X \in \{0,1\}^{30\times 30}$, are relaxed to the continuous domain, $[0,1]^{30\times 30}$, to convert the integer programming problem into a convex form. In each iteration, if an image point $\mathbf{q}_j$ is outside a template point $\mathbf{p}_i$'s trust region, we force $X_{i,j} = 0$. For instance, if image points, $\mathbf{q}_2$, $\mathbf{q}_3$, $\mathbf{q}_4$ are outside $\mathbf{p}_2$'s trust region, we fix the values of the entries, $X_{2,2}$, $X_{2,3}$, and $X_{2,4}$, to 0. To recover point-to-point correspondences between each pair of point sets, we use the method in (18) with $w_h = 0$.

Three state-of-the-art methods, Balanced Graph Matching [8], the learning-based graph matching method [6] and the LP with a locally affine invariant method [19] were used for comparison. Fig. 12 shows the percentages of wrong matchings for our method and other compared ones. The horizontal axis is the frame separation number and the vertical axis is the percentage of wrong matchings. Our method with locally affine model I results in 0.0 percent matching errors for both sequences and outperforms all other methods on the two data sets. Some methods [27], [28] utilized a slightly different experimental setup. Fifteen frames (every seven frames) are sampled from the *Hotel* sequence, which gives 105 pairs of images to match. The two methods [28], [27] reported mis-matching percentages of 0.19 and 4.44 percent, respectively. Our method also achieves 0.0 percent errors with this experimental setup.

The mis-matching rates obtained by our method without the one-to-one constraints are also included in Fig. 12. The error rates are very high because the feature points in the two data sets are very sparse (only 30 points). When trust regions of some template points are shrunk to include only three image points. The dissimilarity functions of the template points become "planes". The template points would thus favor being matching to the input image boundary and result in unsatisfactory matching results. Therefore, the one-to-one constraints are necessary for this experiment. Note that such constraints are also used in [19] for the same data sets. For real-world applications, a large number of feature points would be detected in natural images. Including a few "plane-shape" dissimilarity functions does not affect the matching results much.

5.3 Static Image Pairs

We obtained six static image pairs from [20] and matched them using SIFT [22] points and our locally affine transformation model I. The first five cases in Fig. 13 are surfaces undergoing perspective or very complex local deformations. By approximating them using the locally affine model I, our method satisfactorily matched the first four cases, with some small errors near the boundaries of surfaces where features are more degenerated. In the fifth (flag) case, the left half of the flag was matched satisfactorily but the right half was not because its very large deformation adversely affected the performance of SIFT features. The last case in Fig. 13 shows matching an object in a blurry image to its instance after some deformations in a sharp input image. Both the body and the belt parts of the object were successfully matched although they have undergone large deformations.

5.4 Objects Undergoing Global and Locally Affine Transformations

We captured two video clips (*Spectrum* and *Computer*) by ourselves and obtained two clips (*honeybee* and *butterfly*) from the internet. The object templates were matched to the instances of objects in the videos frame by frame. Fig. 14a shows the four object templates, and Fig. 14c shows example matching results from the four videos.

For different video clips, we chose the most appropriate optimization model from our four transformation models, which best describes the object's transformation and

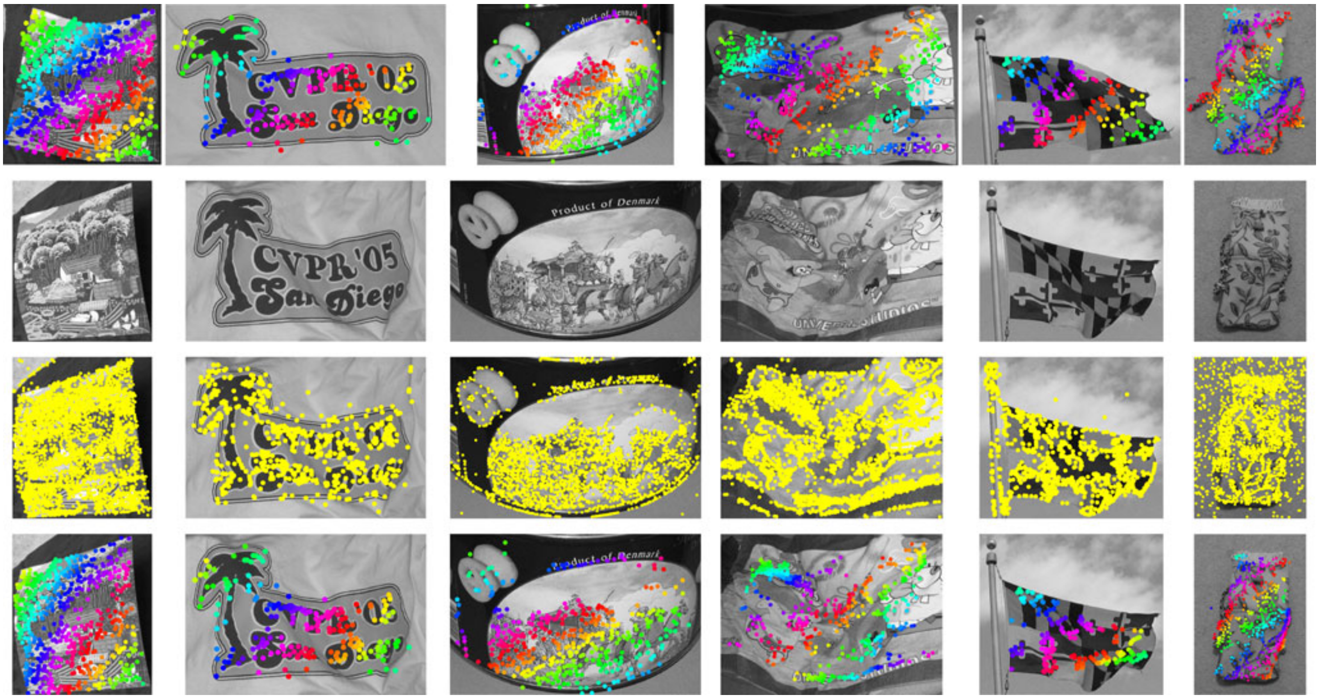


Fig. 13. Experiments on static image pairs. (From left to right) six different matching cases. (First row) Template feature points. (Second row) Input images. (Third row) All feature points detected in the input images. (Fourth row) Matching results on input images using the locally affine model I.

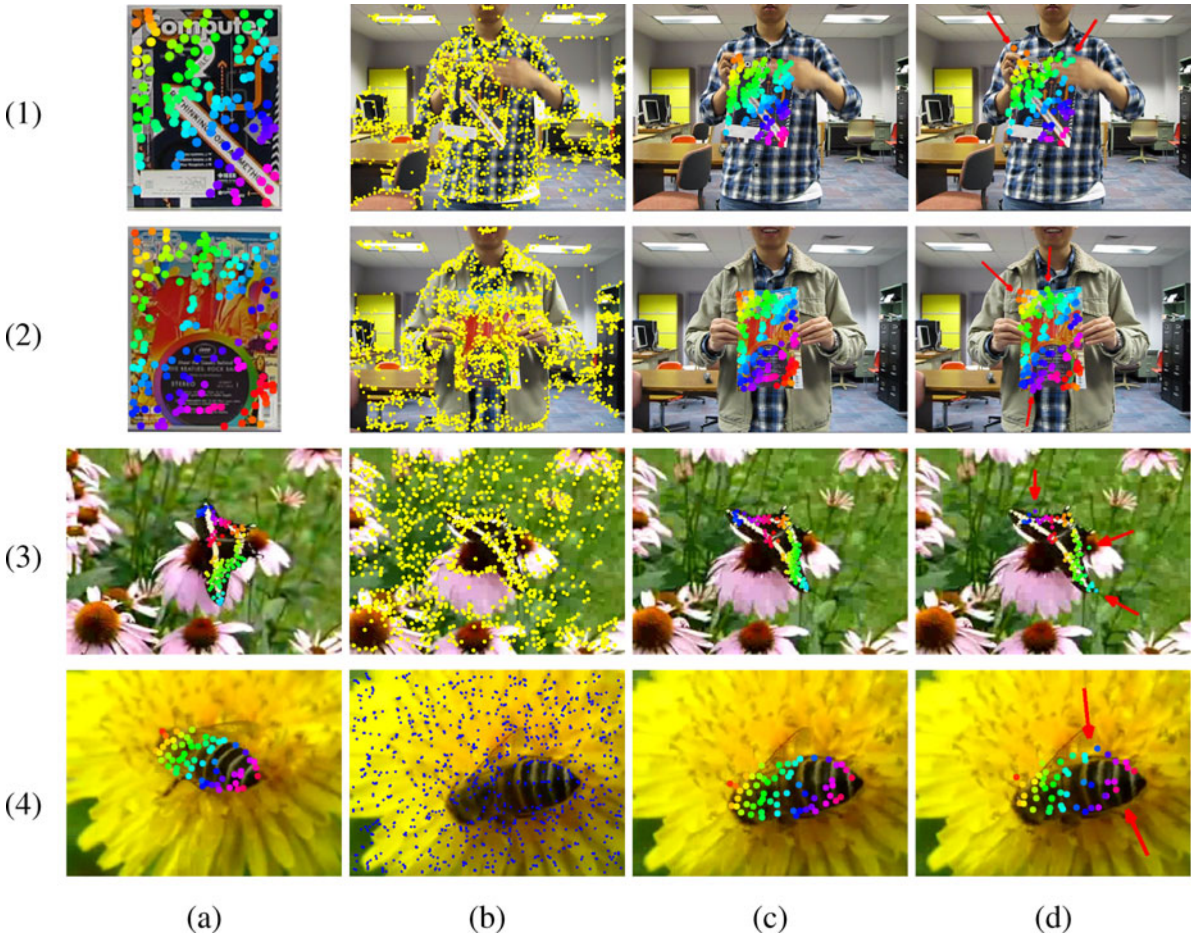


Fig. 14. Example matching results for the four videos by our proposed method and the method in [19]. (a) Template feature points. (b) Input images. Dots represent all detected feature points. (c) The matching results by our proposed method. (d) The matching results by the method in [19].

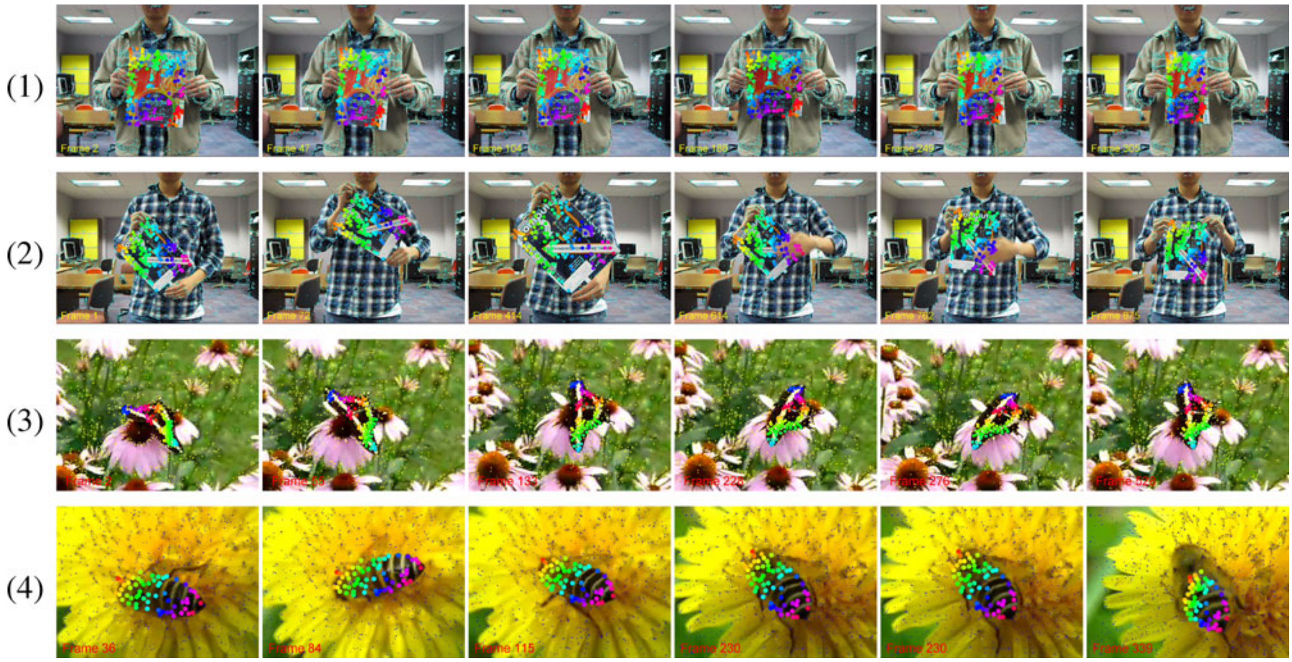


Fig. 15. Sample matching results by our method from (1) the *Spectrum* magazine sequence, (2) the *Computer* sequence, (3) the *butterfly* magazine sequence, and (4) the *honeybee* sequence. All detected feature points are marked in blue in the background.

provides as many geometric constraints as possible. The method in [19] was used for comparison. The first object (Fig. 14(1)) is an IEEE Spectrum magazine mainly undergoing global transformation. We chose to model its transformation using the global affine + local translation model. The second object is a computer magazine undergoing mostly similarity transformation in the first half of the video and local deformations in the second half. We chose to use the locally affine model I to match this object. The third and fourth objects are a butterfly and a honeybee working on a flower. The real-world objects undergo complex transformations, and therefore we again used the locally affine model I to approximate their transformations. Figs. 14c and 14d show the comparison between our results and the results obtained by [19]. It is obvious that our results are more robust to feature point mis-detection. Sample matching results of the four video clips are shown in Fig. 15.

5.5 Planar Surfaces with Given Template Meshes

We also obtained six video clips (*cloth*, *cloth-folds*, *tshirt*, *cushion*, *bed-sheet*, and *paper-bend*) recording different planar surfaces undergoing complex transformations from [25]. Meshes describing these template surfaces' shapes are provided in the data (Fig. 16). Correspondences between the template surfaces (Fig. 16) and input images (Fig. 17) can be used to recover planar surfaces' 3D shapes by algorithms like [25]. We propose to utilize the locally affine model II to solve this matching problem and match the template surfaces to their instances in the video frame by frame. The *cushion*, *bed-sheet*, and *paper-bend* were satisfactorily matched with small errors near boundaries of the surfaces. Some larger matching errors were observed in some frames of the *cloth*, *cloth-folds*, and *tshirt*, but most frames were satisfactorily matched. Sample matching results of the six sequences are shown in Fig. 17. The results

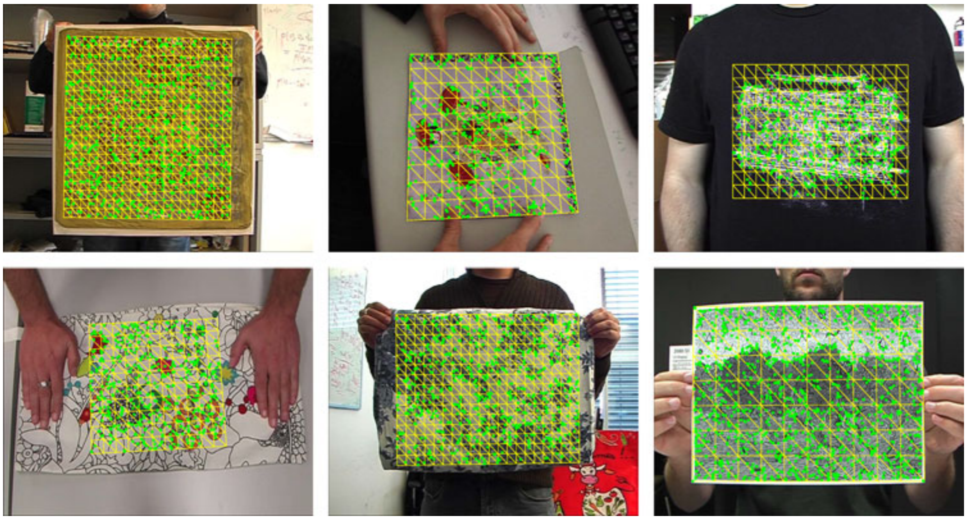


Fig. 16. Template feature points (green dots) and pre-given meshes (yellow lines) for six planar surfaces.

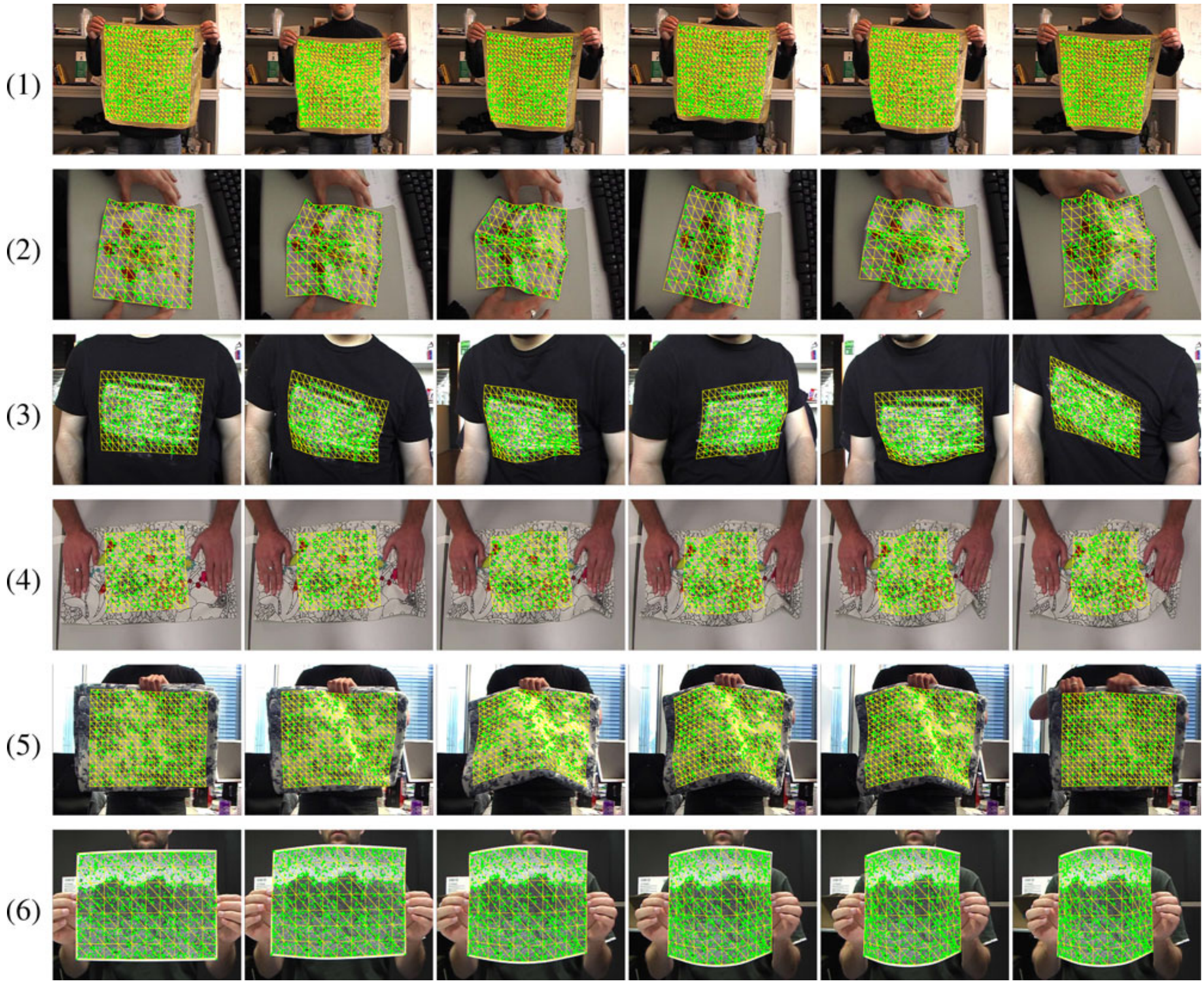


Fig. 17. Sample matching results by our method from (1) the *cloth* sequence, (2) the *cloth-folds* sequence, (3) the *tshirt* sequence, (4) the *cushion* sequence, (5) the *bed-sheet* sequence and (6) the *paper-bend* sequence. Transformed meshes and transformed template points are drawn as yellow lines and green dots, respectively.

demonstrate the potential of our method to recover 3D shapes of planar surfaces from single images.

5.6 Articulated Objects

We created another video by ourselves, which records a toy worm being bent by a person. We chose to use the articulated object's transformation model (16) to approximate its complex transformations. We modeled the toy worm as a template consisting of four connected parts undergoing separate affine transformations. Fig. 18 shows template feature points belonging to the four parts. For better visualization, we only show the four parts' bounding boxes in the results. Fig. 19 shows four example matching results from the video sequence. Although the toy undergoes very complex transformations, our proposed articulated object's model (16) was able to match it satisfactorily.

6 Discussion and Conclusion

6.1 Other Transformation Models

As we mentioned in Section 1.2, other transformation models in the affine-function family, such as the TPS model and the FFD model, can also be used in our proposed framework.

6.2 Occlusion Handling

In Section 5.1, we showed that our method's performance degrades dramatically when a large portion of template points' corresponding image points are occluded. One

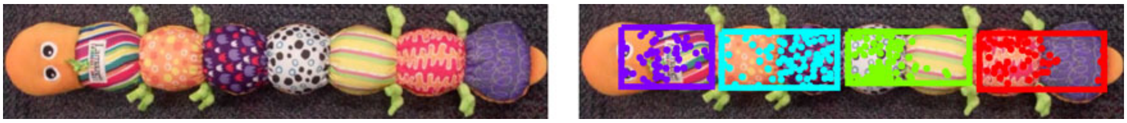


Fig. 18. (Left) The template image of the toy worm. (Right) Four parts' template feature points are colored differently. Boxes are drawn for better visualization in the matching results.

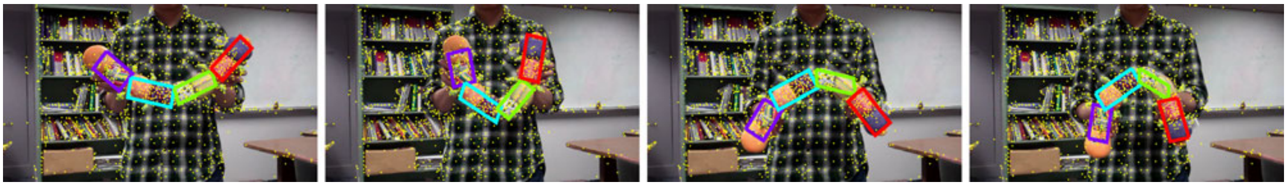


Fig. 19. Four example frames' matching results from the toy worm video by our proposed method. Yellow dots on the background represent all detected feature points.

possible solution is an EM-like scheme similar to that of [30]. We can set a weight w_i for each template point $\mathbf{p}_i$ to represent its distinctiveness. The objective function (2) then becomes

$$\sum_{i=1}^{n_t} w_i c_i(T_i(\Theta_i)) + R(\Theta_1, \ldots, \Theta_m). \tag{20}$$

The weights and the transformation parameters are then optimized iteratively and alternatively. In the first iteration, all template points' weights are fixed and set to the same value. Transformation parameters are optimized as described in Section 4. In the second iteration, each point $\mathbf{p}_i$'s transformation parameters $\Theta_i^{(1)}$ obtained from the first iteration are fixed. w_i is decreased if the ith transformed point $T_i(\Theta_i^{(1)})$'s dissimilarity value $c_i(T_i(\Theta_i^{(1)}))$ is high, and is increased if $c_i(T_i(\Theta_i^{(1)}))$ is low. Similar operations are performed in latter iterations until the change of transformation parameters fall below a pre-given threshold.

6.3 Trust-Region Sizes

In our experiments, we empirically chose trust regions' sizes at each iteration. Developing a systematic way to choose them might be a future research direction.

In summary, we presented a unified feature matching framework which supports the affine-function family of transformation models. For each template point, a convex feature dissimilarity function is created by relaxing its original discrete feature dissimilarity function. The composition of such a convex function with any transformation model in the affine-function family has an equivalent convex optimization form. We proposed four transformation models in this family to solve different matching problems. By solving each template point's transformation parameters explicitly, we can better constrain objects' transformations based on some prior knowledge. Experiments on different transformation models and comparison with previous methods demonstrate the flexibility of our proposed framework.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61301269), in part by the PhD Programs Foundation of Ministry of Education of China (No. 20130185120039), in part by Sichuan Provincial Key Technology Research and Development Program (No. 2014GZX0009), in part by the Fundamental Research Funds for the Central Universities (No. 2672014ZYGX2013J017), and in part by China Postdoctoral Science Foundation (No. 2014M552339).

References

[1] CMU hotel data set. (2006) [Online]. Avaialble: http://vasc.ri.cmu.edu/idb/html/motion/hotel/index.html

[2] CMU house data set. (2006) [Online]. Available: http://vasc.ri.cmu.edu/idb/html/motion/house/index.html

[3] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.

[4] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, pp. 26–33.

[5] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 500–513, Mar. 2011.

[6] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1048–1058, June 2009.

[7] S. Choi, T. Kim, and W. Yu, "Performance evaluation of RANSAC family," in *Proc. Brit. Mach. Vis. Conf.*, 2009, pp. 1–12.

[8] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in *Proc. Neural Inf. Process. Syst.*, 2006, pp. 313–320.

[9] A. D. J. Cross and E. R. Hancock, "Graph matching with a dual-step EM algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1236–1253, Nov., 1998.

[10] O. Duchenne, F. Bach, I. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1980–1987.

[11] O. Duchenne, A. Joulin, and J. Ponce, "A graph-matching kernel for object categorization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1792–1799.

[12] P. Felzenszwalb and D. Huttenlocher, "Efficient matching of pictorial structures," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2000, vol. 2, pp. 66–73.

[13] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[14] H. Jiang, M. S. Drew, and Z. Li, "Matching by linear programming and successive convexification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 959–975, Jun. 2007.

[15] H. Jiang and S. X. Yu, "Linear solution to scale and rotation invariant object matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 2474–2481.

[16] H. Jiang, T.-P. Tian, and S. Sclaroff, "Scale and rotation invariant matching using linearly augmented trees," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 2473–2480.

[17] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, pp. 1482–1489.

[18] M. Leordeanu and M. Hebert, "Unsupervised learning for graph matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 864–871.

[19] H. Li, X. Huang, and L. He, "Object matching using a locally affine invariant and linear programming techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 411–424, Feb. 2013.

[20] H. Ling and D. W. Jacobs, "Deformation invariant image matching," in *Proc. Int. Conf. Comput. Vis.*, 2005, pp. 1466–1473.

[21] H. Liu and S. Yan, "Common visual pattern discovery via spatially coherent correspondences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1609–1616.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

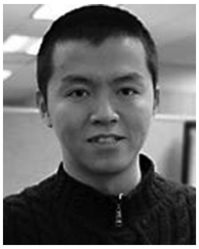
[23] N. Payet and S. Todorovic, "From contours to 3D object detection and pose estimation," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 983–990.

[24] R. Raguram, J. Frahm, and M. Pollefeys, "A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus," in *Proc. European Conf. Comput. Vis.*, vol. 2, 2008, pp. 500–513.
[25] M. Salzmann and P. Fua, "Linear local models for monocular reconstruction of deformable surfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 931–944, May 2011.
[26] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 5, pp. 530–535, May 1997.
[27] M. Torki and A. Elgammal, "One-shot multi-set non-rigid feature-spatial matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 3058–3065.
[28] L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 596–609.
[29] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1744–1757, Sep. 2012.
[30] J. Yang, R. S. Blum, J. P. Williams, Y. Sun, and C. Xu, "Non-rigid image registration using geometric features and local salient region features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1, 2006, pp. 825–832.
[31] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
[32] Y. Zheng, A. A. Hunter, J. Wu, H. Wang, J. Gao, M. G. Maguire, and J. C. Gee, "Landmark matching based automatic retinal image registration with linear programming and self-similarities," in *Proc. Int. Conf. Inform. Process. Med. Imag.*, 2011, pp. 674–685.

Hongsheng Li received the bachelor's degree in automation from East China University of Science and Technology, and the master's and doctorate degrees in computer science from Lehigh University, Pennsylvania, in 2006, 2010, and 2012, respectively. He is an associate professor in the School of Electronic Engineering at University of Electronic Science and Technology of China. His research interests include computer vision, medical image analysis, and machine learning.

Xiaolei Huang received the bachelor's degree in computer science and engineering from Tsinghua University, China, in 1999, and the master's and doctorate degrees in computer science from Rutgers, The State University at New Brunswick, New Jersey, in 2001 and 2006, respectively. She is currently an associate professor in the Computer Science and Engineering Department at Lehigh University, Bethlehem, Pennsylvania. Her research interests are in the areas of computer vision, biomedical image analysis, computer graphics, and multimedia retrieval. She serves on the program committees of several international conferences on computer vision, biomedical image computing, and computer graphics, and she is a regular reviewer for journals including the *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, and the *IEEE Transactions on Medical Imaging (TMI)*. She is a member of the IEEE and the IEEE Computer Society.

Junzhou Huang received the BE degree from Huazhong University of Science and Technology, Wuhan, China, in 1996, the MS degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2003, and the PhD degree from Rutgers University, New Brunswick, New Jersey, in 2011. He is an assistant professor in the Computer Science and Engineering Department at the University of Texas at Arlington. His research interests include biomedical imaging, machine learning, data mining and computer vision, with focus on the development of sparse modeling, imaging and learning techniques for massive data problems. He is a member of the IEEE.



Shaoting Zhang received the BE degree from Zhejiang University in 2005, the MS degree from Shanghai Jiao Tong University in 2007, and the PhD degree in computer science from Rutgers in January 2012. He is an assistant professor in the Department of Computer Science at the University of North Carolina at Charlotte. Before joining UNC Charlotte, he was a faculty member in the Department of Computer Science at Rutgers-New Brunswick (Research Assistant Professor, 2012-2013). His research is on the interface of medical imaging informatics, visual understanding and machine learning. He is a member of IEEE.