# Optimal Object Matching via Convexification and Composition

Hongsheng Li[1] Junzhou Huang[2] Shaoting Zhang[3] Xiaolei Huang[1]
[1] Department of Computer Science & Engineering, Lehigh University, USA
[2] Department of Computer Science & Engineering, University of Texas at Arlington, USA
[3] Department of Computer Science, Rutgers University, USA
h.li@lehigh.edu, jzhuang@uta.edu, shaoting@cs.rutgers.edu, xih206@lehigh.edu

## Abstract

*In this paper, we propose a novel object matching method to match an object to its instance in an input scene image, where both the object template and the input scene image are represented by groups of feature points. We relax each template point's discrete feature cost function to create a convex function that can be optimized efficiently. Such continuous and convex functions with different regularization terms are able to create different convex optimization models handling objects undergoing (i) global transformation, (ii) locally affine transformation, and (iii) articulated transformation. These models can better constrain each template point's transformation and therefore generate more robust matching results. Unlike traditional object or feature matching methods with "hard" node-to-node results, our proposed method allows template points to be transformed to any location in the image plane. Such a property makes our method robust to feature point occlusion or mis-detection. Our extensive experiments demonstrate the robustness and flexibility of our method.*

## 1. Introduction

An object template in an image can be represented by a group of template feature points (Fig. 1.(a)). Each feature point has a location $(x, y)$ in the 2D image domain and a feature vector describing the local appearance around that location. Object matching can be referred as locating such a template in an input scene image represented by thousands of detected feature points (Fig. 1.(b)). It has extensive uses in object classification [1], detection and tracking [8], shape matching [8], and image retrieval [15].

Most feature point matching methods aim at recovering node-to-node correspondences between two feature point sets. The family of RANSAC methods [3] has shown its effectiveness in various matching tasks. However, these methods are restricted to matching objects undergoing only global transformations. Graph matching methods are proposed to handle more complex local transformations, where feature points are modeled as graph nodes, and geometric relations between pairs of feature points are modeled as graph edges. Berg *et al*. [1] modeled the problem as a quadratic integer programming model, where affine terms and quadratic terms of the objective function represent node-to-node and edge-to-edge similarities between the two graphs, respectively. The model was relaxed into a continuous domain, solved, and mapped back into the original solution space. Leordeanu and Hebert [9] encoded all the above similarity information into a matrix. The correspondences are then 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 similarity matrix, supervised [2] and unsupervised [10] learning methods optimizing the weights were proposed. Cour *et al*. [4] 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. Most recently, Liu and Yan [13] proposed an algorithm to discover all common visual patterns within two sets of feature points. It optimizes the same objective function as that of [9] but with different constraints. It showed its effectiveness in recovering visual common patterns no matter the matchings between them are one-to-one or many-to-many. However, one major limitation of the graph matching methods is that order-2 edges can only provide rotational invariance. Zass and Shashua [16] extended ordinary graphs to hypergraphs, whose high-order edges can encode more complex geometric invariances. The method's output is a probabilistic ("soft") result rather than traditional "hard" node-to-node results. In this way, they were able to

Figure 1. (Left) A group of feature points representing the object *template*. (Right) Thousands of feature points representing the input *scene* image.

model the problem as a convex optimization problem and obtained a global minimum. Duchenne *et al*. [5] encoded similarities between two hypergraphs into a tensor and proposed a power iteration method to effectively recover the tensor's principal eigenvector with sparse prior.

The feature point matching problem is also modeled as linear programming models, where geometric invariances are expressed as affine functions and constraints. Jiang *et al*. [7] proposed a linear programming framework for feature point matching. It models each node-to-node correspondence as a binary variable and uses vectors specified by each point and its neighbors as geometric invariances. Such geometric invariance terms can only tolerate global translations and local deformations. Jiang and Yu [8] followed this framework and explicitly solved for rotation and scaling to achieve similarity invariance. Li *et al*. [11] represented each template point by an affine combination of its neighbors. Locally-affine invariance can be obtained by reconstructing matched points using the same affine combination coefficients. One disadvantage of this type of methods is that their geometric regularization terms must be affine functions and cannot be easily combined. Therefore, this framework would have difficulties modeling objects undergoing complex transformations, *e.g*., articulated transformation.

In this paper, we propose a novel object matching method. For each template feature point, we create a convex matching function based on its feature similarities to all scene points. Based on such convex functions, we propose a convex optimization framework and solve for all points' globally optimal transformation parameters simultaneously. Compared with existing feature matching methods, it has two major advantages. (i) Our proposed framework can provide multiple levels of degrees of freedom on transformation and therefore can better constrain transformations of different types of object templates. For instance, if an object undergoes articulated transformation, we can then explicitly model a template consisting of moving rigid parts and junction points connecting these parts. (ii) Unlike traditional methods seeking "hard" node-to-node results, our new method no longer requires each template point being matched to only scene feature point locations but the entire image plane. This "soft" result can provide more robust matching performance in real world applications where some corresponding feature points in the scene image might not be detected. However, when a "hard" result is desired, our method still can obtain it through a subsequent step.

# 2. Feature Cost Functions

## 2.1. Feature Matching Costs

As we mentioned above, both the template and the input scene image can be represented by groups of feature points (Fig. 1). Let $n_t$ and $n_s$ represent the number of feature points in the template and in the input scene image, respectively. Let $\mathbf{p}_i = [x_{\mathbf{p}_i}, y_{\mathbf{p}_i}]^T, i = 1, \cdots, n_t$, and $\mathbf{q}_j = [x_{\mathbf{q}_j}, y_{\mathbf{q}_j}]^T, j = 1, \cdots, n_s$, denote the position of the $i$th feature point in the template, and the position of $j$th feature point in the scene image, respectively.

Usually, feature vectors are invariant to certain geometric transformation, *e.g*., SIFT [14] feature vectors are rotational and scaling invariant. The feature cost of matching two such feature points can be calculated as the $L_2$ distance between their feature vectors. Even if some type of features does not have geometrical invariance (*e.g*., Shape Context [1]), one can still exhaustively search for all geometric transformations for one feature vector and calculate the $L_2$ distances between the transformed feature vector and another one. The matching cost between the two feature points can then be defined as the minimum among all the $L_2$ distances. Let $C_{i,j}$ denote the feature cost of matching the $i$th template feature point to the $j$th scene feature point. Such discrete feature costs are all pre-calculated before the matching is performed.

## 2.2. Discrete Feature Cost Functions

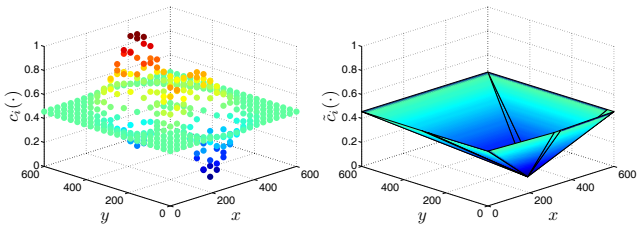
For each template point $\mathbf{p}_i, i = 1, \cdots, n_t$, we can define a discrete feature cost function $c_i : Q \rightarrow \mathbf{R}$ as follows:

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

where $Q = \{\mathbf{q}_1, \cdots, \mathbf{q}_{n_s}\}$ is the set containing all scene points' positions. This function denotes that the template point $\mathbf{p}_i$ can only be matched to scene feature point locations, $\mathbf{q}_j = [x_{\mathbf{q}_j}, y_{\mathbf{q}_j}]^T, j = 1, \cdots, n_s$, with a feature cost determined by function $c_i(\cdot)$. This is because only the feature costs of matching $\mathbf{p}_i$ to scene points' positions are defined as $C_{i,j}$, $j = 1, \cdots, n_s$. Minimization of $c_i(\cdot)$ results in the best matched scene point for the $i$th template point $\mathbf{p}_i$. One example of this discrete function is shown in Fig. 2.(a). Those $c_i(\cdot)$ functions are discrete and non-convex. Directly optimizing the summation of a series of $c_i(\cdot)$ functions with geometric regularization terms is NP-hard, and no algorithm can optimize it in polynomial time.

## 2.3. Convex Feature Cost Functions

To solve this problem, we relax each $c_i(\cdot)$ function and create a continuous and convex feature cost function $\tilde{c}_i(\cdot)$ which can be efficiently optimized. The above discrete functions are viewed as 3D point clouds. For the template point $\mathbf{p}_i$, all scene feature points' locations, and the costs of matching $\mathbf{p}_i$ to all scene points can be viewed as a 3D ($n_s \times 3$) point cloud (Fig. 2.(a)), where the 3rd dimension



(a) (b)

Figure 2. (a) An example discrete feature cost function $c_i(\cdot)$. (b) The convex feature cost function $\tilde{c}_i(\cdot)$ obtained from (a).

represents the matching cost:

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

One way to create the convex feature cost function $\tilde{c}_i(\cdot)$ is to define it as the lower convex hull of the $i$th 3D point cloud with respect to the matching cost 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 matching cost dimension). One example of this convex feature function is shown in Fig. 2.(b). Let $n_f$ denote the number of facets on the lower convex hull, and $z_k = r_k x + s_k y + t_k,\ k = 1, \cdots, n_f$, be the plane functions defined by these facets, where $r_k$, $s_k$, and $t_k$ are coefficients of the $k$th plane function. The convex feature cost function $\tilde{c}_i : \mathbf{R}^2 \rightarrow \mathbf{R}$ can be defined as

$$\tilde{c}_i\left([x, y]^T\right) = \max_k \left(r_k x + s_k y + t_k\right). \tag{3}$$

It denotes that the $i$th template point now can be assigned to any location $[x, y]^T$ in the 2D image domain with a matching cost $\tilde{c}_i\left([x, y]^T\right)$. Minimization of $\tilde{c}_i(\cdot)$ no longer matches the $i$th template point to only scene point locations but any location in the 2D image domain. Note that the convex functions $\tilde{c}_i(\cdot)$ are not limited to the lower convex hulls, any other types of convex functions obtained by relaxing the discrete feature cost functions (1) can also be used.

The lower convex hull technique has an intuitive interpretation (Fig. 2). The lower convex hulls are lower bounds of the discrete feature cost functions. However, when features are not distinctive, this technique may not generate satisfactory lower bounds. We will introduce an iterative technique to gradually provide more accurate lower bounds in Section 4.

Instead of directly searching for an optimal matching position for each template point, we prefer deforming template points with some geometric transformation models and determining their feature costs using (3). In this way, we can better constrain all template points' transformations as well as neighboring points' geometric relationships. We deform the template point $\mathbf{p}_i$ with affine transformation

$$T_i^a(\Theta) = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} x_{\mathbf{p}_i} \\ y_{\mathbf{p}_i} \end{bmatrix} + \begin{bmatrix} \phi \\ \varphi \end{bmatrix}, \tag{4}$$

or similarity transformation

$$T_i^s(\Theta) = \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} \begin{bmatrix} x_{\mathbf{p}_i} \\ y_{\mathbf{p}_i} \end{bmatrix} + \begin{bmatrix} \phi \\ \varphi \end{bmatrix}, \tag{5}$$

where $T_i^a : \mathbf{R^6} \rightarrow \mathbf{R^2}$ and $T_i^s : \mathbf{R}^4 \rightarrow \mathbf{R^2}$ denote the affine and similarity transformations of the template point $\mathbf{p}_i$ with parameters $\Theta$ respectively. $\Theta = [\alpha, \beta, \gamma, \delta, \phi, \varphi]^T \in \mathbf{R}^6$ represent affine transformation parameters, and $\Theta = [\alpha, \beta, \phi, \varphi]^T \in \mathbf{R}^4$ the similarity transformation parameters. We further denote $T_i(\Theta) = A\mathbf{p}_i + b$ as a generally transformed template point $\mathbf{p}_i$ with parameters $\Theta$. Note that $x_{\mathbf{p}_i}$ and $y_{\mathbf{p}_i}$ are not variables but coefficients of the function $T_i(\Theta)$. The final feature cost of the $i$th transformed template point can then be determined by using its position, $T_i(\Theta)$, as the variables of $\tilde{c}_i(\cdot)$, *i.e.*,

$$f_i(\Theta) = \tilde{c}_i\left(T_i(\Theta)\right). \tag{6}$$

$f_i(\Theta)$ can be viewed as the $i$th template point's feature matching cost under a transformation specified by $\Theta$. Because both (4) and (5) are affine functions, and (3) is a convex function, the composition of them, $f_i(\Theta)$, is also convex.

Consequently, $f_i(\Theta)$ can be efficiently and globally optimized by convex optimization. The gradients of $f_i(\Theta)$ can be easily obtained by the chain rule:

$$\nabla f_i(\Theta) = \nabla \tilde{c}_i(T_i(\Theta)) \nabla T_i(\Theta). \tag{7}$$

### 2.4. The Overall Objective Function

Since each feature cost function $f_i(\Theta)$ is convex, the summation of all feature cost functions, $\sum_i^{n_t} f_i(\Theta)$, is also convex and therefore can be efficiently optimized. The optimal transformation parameters for all points $\mathbf{p}_1, \cdots, \mathbf{p}_{n_t}$ are denoted as $\hat{\Theta}_1, \cdots, \hat{\Theta}_{n_t}$. They can be obtained by optimizing the following overall objective function,

$$\underset{\Theta_1, \cdots, \Theta_{n_t}}{\arg\min} \left( \sum_i^{n_t} f_i(\Theta_i) + \text{Regularization Terms} \right), \tag{8}$$

where $\Theta_i$ denotes $\mathbf{p}_i$'s transformation parameters, and the regularization terms represent geometric constraints that different transformation models should maintain.

One major advantage of the above objective function is that it no longer requires template points being matched to only scene point locations, instead it can be matched to any position in the image plane. For each template point, our method calculates its optimal transformation parameters. This property is very useful when a small part of the

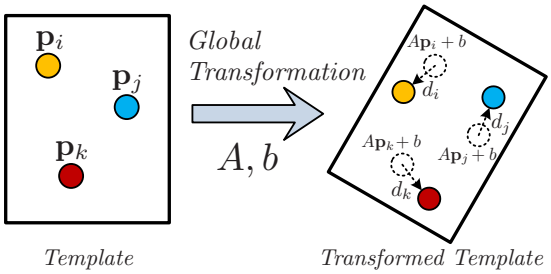



Figure 3. Illustration of the global transformation model. After the global transformation with parameters $A$ and $b$, the three points individually translate for $d_i$, $d_j$, and $d_k$ to better fit the input image.

object is occluded in the scene image. Those occluded feature points can be "interpolated" from the unoccluded ones. Even if the object is not occluded, this property is still useful because some corresponding feature points might not be detected in the scene image.

However, if a "hard" node-to-node result is desired, we can search for the best matched scene point for the template point $\mathbf{p}_i$ after its optimal transformation parameters $\hat{\Theta}_i$ is obtained from (8):

$$\hat{j}_i = \arg\min_j \left( c_i\left(\mathbf{q}_j\right) + w_h \left\| \mathbf{q}_j - T_i(\hat{\Theta}_i) \right\|_2^2 \right), \quad (9)$$

The above function means the best matched scene point $\mathbf{q}_{\hat{j}_i}$ of $\mathbf{p}_i$ should have a small feature matching cost as well as be close to the optimal "soft" matching result obtained from (8). $w_h$ is the parameter that weighs the feature matching cost and the squared distance to the optimal "soft" result.

## 3. Transformation Models

For objects undergoing different transformations, different regularization terms should be used in (8). In this section, we introduce three different transformation models with multiple levels of degrees of freedoms: (i) the global transformation model, (ii) the locally affine transformation model, and (iii) the articulated transformation model. Note that transformation models are not limited to the ones we mention here, combinations of the above models or other models can also be used in our proposed framework.

Unlike existing methods, where geometric constraints are implicitly expressed as edge weight differences [1], [9], [16], [5] or affine functions [7], [8], [11], our method explicitly optimizes and constrains each template point's transformation parameters. Therefore, it can better constrain geometric relationships between transformed points.

### 3.1. The Global Transformation Model

Our global transformation model assumes objects undergoing globally affine or similarity transformation, and small local deformations. Therefore, all points should share a common set of global transformation parameters. To model the local deformations and to better fit the input image

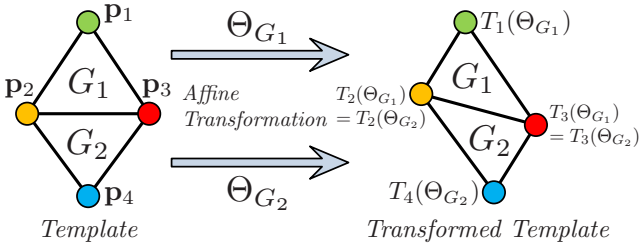



Figure 4. Illustration of the locally affine transformation model using 4 points ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$) in 2 triangles ($G_1$ and $G_2$). To maintain the mesh topology after transformation, the equality constraints (12) on $\mathbf{p}_2$ and $\mathbf{p}_3$ should be satisfied as $T_2(\Theta_{G_1}) = T_2(\Theta_{G_2})$, and $T_3(\Theta_{G_1}) = T_3(\Theta_{G_2})$.

locally, each point is also allowed to translate individually for a small distance. We make small changes to the $T_i(\Theta)$ function and transform the template point $\mathbf{p}_i$ as

$$\dot{T}_i(\Theta_i) = A\mathbf{p}_i + b + d_i, \quad (10)$$

where $A$ represents the $2 \times 2$ global transformaton matrix, and $b$ the $2 \times 1$ global translation vector as in (4) and (5). $d_i = [\phi_i, \varphi_i]^T$ is $\mathbf{p}_i$'s local translation vector. The template point $\mathbf{p}_i$'s transformation parameters $\Theta_i$ consist of common global transformation parameters, $A$ and $b$, and a local translation vector $d_i$ (Fig. 3). To penalize local deformations that are too large, the squared distance each point translates locally, $\|d_i\|_2^2$, should be regularized. The above transformation model and regularization terms result in the following optimization model

$$\underset{A,b,d_1,\cdots,d_{n_t}}{\text{minimize}} \sum_i^{n_t} \left\{ \tilde{c}_i(A\mathbf{p}_i + b + d_i) + w_g \|d_i\|_2^2 \right\}, \quad (11)$$

where $w_g$ is the parameter that weighs the feature cost terms and the local translation regularization terms. The above optimization model provides more robust matching results when the object is known to undergo mostly global transformation. Experiments demonstrating this model are shown in Section 5.1 and 5.3.

### 3.2. The Locally Affine Transformation Model

If an object undergoes complex transformations that cannot be described by globally affine or similarity transformation models, we propose to approximate the object's transformation with a locally affine transformation model.

This model achieves locally affine invariance by transforming every three neighboring template points together. We first use Delaunay Triangulation to obtain a triangulated mesh from the 2D template points. Then each three points defining a triangle on the mesh are transformed together, in other words, every three template points in a triangle share a common set of affine transformation parameters (Fig. 4). Let $m$ denote the number of triangles in the triangulated mesh, $G_1, G_2, \cdots, G_m$ denote the $m$ sets consisting of the points in the 1st, 2nd, $\cdots$, $m$th triangles, and

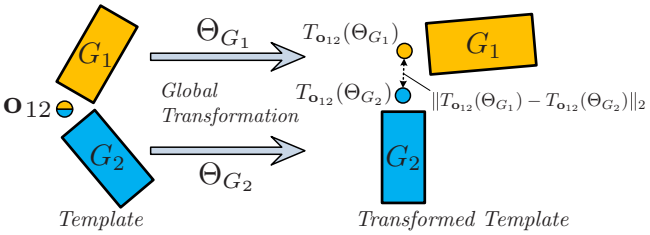



Figure 5. Illustration of the articulated transformation model with a 2-part object. $\mathbf{o}_{12}$ is the junction point between parts $G_1$ and $G_2$. The distance between the transformed junction points $T_{\mathbf{o}_{12}}(\Theta_{G_1})$ and $T_{\mathbf{o}_{12}}(\Theta_{G_2})$ is used for regularization.

$\Theta_{G_u} \in \mathbf{R}^6$ denote the affine transformation parameters for template points in the $u$th triangle. If the $u$th and the $v$th triangles share a common edge, we call them two neighboring triangles and denote them as $u \in \mathcal{N}_v, v \in \mathcal{N}_u$.

To make sure one template point in several triangles being transformed to a single position, the following equality constraints need to be added in the optimization model:

$$\begin{aligned} T_i(\Theta_{G_u}) = T_i(\Theta_{G_v}) \quad & \text{for all } i = 1, \cdots, n_t, \\ & \text{for all } u, v = 1, \cdots, m, \\ \text{where } \mathbf{p}_i \in G_u \text{ and } & \mathbf{p}_i \in G_v, \end{aligned} \tag{12}$$

The above equality constraints are illustrated in Fig. 4 with 4 points, $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$, in two triangles, where $G_1 = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ 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 $T_2(\Theta_{G_1}) = T_2(\Theta_{G_2}), T_3(\Theta_{G_1}) = T_3(\Theta_{G_2})$.

We also would like the transformed mesh to 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

$$\begin{aligned} \underset{\Theta_{G_1}, \cdots, \Theta_{G_m}}{\text{minimize}} \quad & \left( \sum_{u=1}^{m} \sum_{\mathbf{p}_i \in G_u} \tilde{c}_i(T_i(\Theta_{G_u})) \right. \\ & \left. + w_l \sum_{u=1}^{m} \sum_{u \in \mathcal{N}_v} \|\Theta_{G_u} - \Theta_{G_v}\|_2^2, \right) \\ \text{subject to} \quad & \textit{The Equality Constraints in (12)}, \end{aligned} \tag{13}$$

where $w_l$ is the parameter that weighs the feature cost terms and the mesh smoothness regularization terms. The above model can approximate very complex transformations. Experiments demonstrating this model are shown in Section 5.2 and 5.3.

### 3.3. The Articulated Transformation Model

There are also many types of objects, *e.g.*, human bodies, undergoing articulated transformations. These objects have several connected rigid parts which undergo global transformations separately. We reuse the notations in Section 3.2 without causing confusion. Let $m$ denote the number of rigid parts in an object, $G_1, G_2, \cdots, G_m$ denote the $m$ sets consisting of the points in the 1st, 2nd, $\cdots$, $m$th parts, and $\Theta_{G_u} \in \mathbf{R}^6$ or $\mathbf{R}^4$ denote the affine or similarity transformation parameters for template points in the $u$th part.

For every pair of connected parts $u$ and $v$, we model a junction point $\mathbf{o}_{uv}$ denoting the connecting point between the two parts. During the matching process, it is deformed according to both parts' transformation parameters to $T_{\mathbf{o}_{uv}}(\Theta_{G_u})$ and $T_{\mathbf{o}_{uv}}(\Theta_{G_v})$, where $T_{\mathbf{o}_{uv}}(\Theta)$ is the transformed position of $\mathbf{o}_{uv}$ after a transformation with parameters $\Theta$ and is defined similarly as $T_i(\Theta)$. To maintain connectivity of the two parts $u$ and $v$, the squared distance between the two transformed junction points, $\|T_{\mathbf{o}_{uv}}(\Theta_{G_u}) - T_{\mathbf{o}_{uv}}(\Theta_{G_v})\|_2^2$, should be minimized and is used as a regularization term. The articulated transformation model for a two-part object is illustrated in Fig. 5. The final optimization model for an object undergoing generally articulated transformation is then defined as

$$\begin{aligned} \underset{\Theta_{G_1}, \cdots, \Theta_{G_m}}{\text{minimize}} \quad & \left( \sum_{u=1}^{m} \sum_{\mathbf{p}_i \in G_u} \tilde{c}_i(T_i(\Theta_{G_u})) \right. \\ & \left. + w_a \sum_{\mathbf{o}_{uv}} \|T_{\mathbf{o}_{uv}}(\Theta_{G_u}) - T_{\mathbf{o}_{uv}}(\Theta_{G_v})\|_2^2 \right), \end{aligned} \tag{14}$$

where $w_a$ is the parameter that weighs the feature cost terms and the part connectivity regularization terms. Experiments demonstrating this model are shown in Section 5.4.

## 4. Implementation and Performance

The convex feature cost functions $\tilde{c}_i(\cdot)$ are created by relaxing the discrete functions $c_i(\cdot)$. When features are distinctive, *i.e.*, most template points have relatively low costs when being matching to their corresponding scene 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 "large" low-cost regions. To solve this problem, we use a technique similar to that proposed in [7]. We iteratively create these convex feature functions using fewer and fewer scene feature points and gradually obtain more accurate lower bounds.

In the first iteration, for each template point, we use all scene feature points and their feature costs as (2) to create the convex feature cost function. Those convex functions were then used in our proposed optimization models, and template points are transformed according to the obtained optimal transformation parameters. In the second iteration, for each template point, we set a "trust region" centered at the first iteration's transformed template point position. Each trust region is smaller than the entire image and therefore includes fewer scene points. We then can generate feature cost functions using fewer scene points and obtain more accurate lower bounds for the discrete $c_i(\cdot)$ functions. Similar operations are performed in the following iterations,

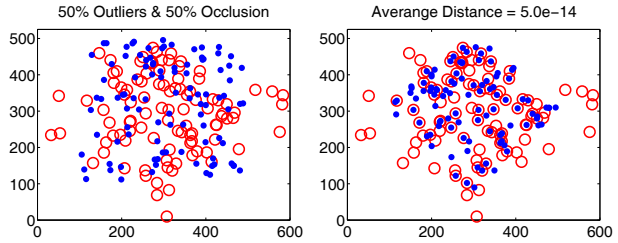


Figure 6. An example point matching case using Shape Context [1] as features (left) before matching and (right) after matching. Blue dots and red circles represent template and scene feature points.

| | Our Proposed Method | The Previous Method [8] |
|---|---|---|
| $h\% = 10\%$ | $\mathbf{0.00 \pm 0.00}$ | $1.34 \pm 4.06$ |
| $h\% = 20\%$ | $\mathbf{0.10 \pm 1.08}$ | $4.37 \pm 11.29$ |
| $h\% = 30\%$ | $\mathbf{0.53 \pm 2.03}$ | $6.59 \pm 13.01$ |
| $h\% = 40\%$ | $\mathbf{3.27 \pm 6.23}$ | $10.23 \pm 13.88$ |
| $h\% = 50\%$ | $\mathbf{18.72 \pm 17.81}$ | $19.94 \pm 21.66$ |

Table 1. The mean and standard deviation of point matching cases' errors under different outlier and occlusion levels. $L_2$ distances between known corresponding points are used as errors.

where feature cost functions are created based on smaller trust regions centered at the previous iterations' resulting transformed template points.

Our optimization models (11), (13), and (14) are asymptotically faster than the previous methods [8], [11] because our models have significantly fewer variables and constraints. For each matching case with a 100-node template, our method needs no more than 3 iterations and each iteration takes no more than $1.5s$ using a MATLAB and CVX [6] implementation on a computer with a 3.0GHz CPU.

## 5. Experiments

In this section, we present extensive experiments to demonstrate the effectiveness and robustness of our method. Except for the experiments in Section 5.1, where Shape Context [1] is used as features, SIFT [14] feature points are used for all other experiments. For each case, we tested three weights, 0.2, 1 and 5 for our method; the weight resulting in the best matching result was chosen. In all our experiments, we let our method run for three iterations. In the 1st iteration, the trust region for each template point is set as the entire image. In the 2nd and 3rd iterations, the trust region is set as a $151 \times 151$ square and a $25 \times 25$ square centered at the previous iterations' results, respectively.

### 5.1. Synthetic Data

For experiments on synthetic data, we slightly changed the experiment setup in [8] and created random-point matching cases with Shape Context [1] as features. 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 scene points from the template points, we randomly scaled the template points in the range $[0.5, 2.0]$ and rotated them for an angle in $[-\pi, \pi]$. Finally, we randomly deleted $h\% \times n_t$ number of points from the template point set to simulate the effects of occlusion or failure of feature point detection, and added $h\% \times n_t$ number of randomly spread points in $[0, 600] \times [0, 600]$ as outliers. The $L_2$ distances between the transformed template points and their known corresponding scene positions are calculated as matching errors for each matching case. One matching case with $50\%$ outliers and $50\%$ occlusion using our method is shown in Fig. 6. For each $10\%$, $20\%$, $30\%$, $40\%$ and $50\%$ outlier and occlusion level, we created 100 matching cases and matched them using our global similarity model (11) and the method in [8]. The statistics of errors on the matching cases using the two methods are compared in Table 1. Our global similarity model (11) provides more constraints than [8] does, and is able to "interpolate" missing template points caused by occlusion or failure of feature point detection.

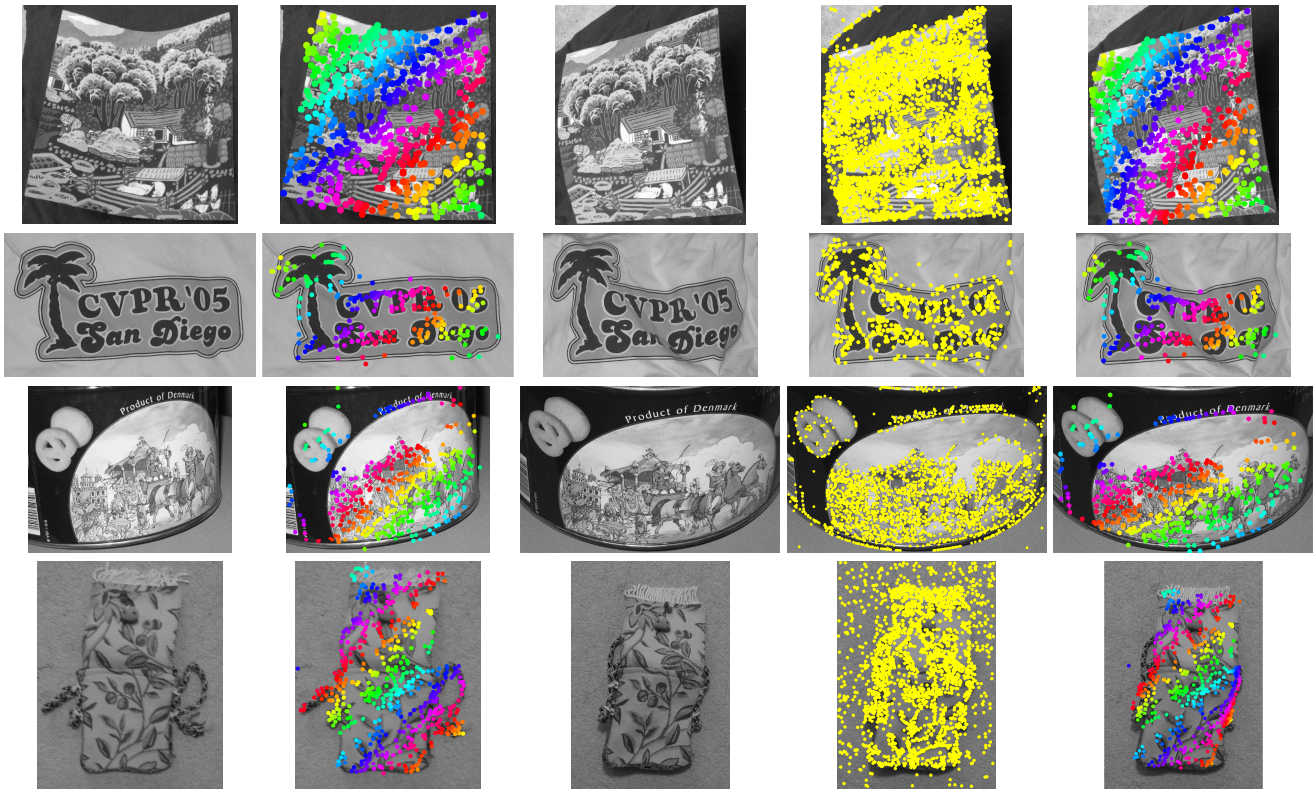
### 5.2. Static Image Pairs

We obtained 4 static image pairs from [12] and matched them using SIFT [14] points and our locally affine optimization model (13). The first three objects in Fig. 7 are surfaces undergoing perspective or very complex local deformations. By approximating them using the locally affine model (13), our method satisfactorily matched them, with some small errors near the boundaries of objects where features are more degenerated. The last case in Fig. 7 shows matching an object in a blurry image to its instance after some deformations in a sharp scene image. The body and belt parts of the object were successfully matched although they have undergone large deformations.

### 5.3. Objects Undergoing Global and Locally Affine Transformations

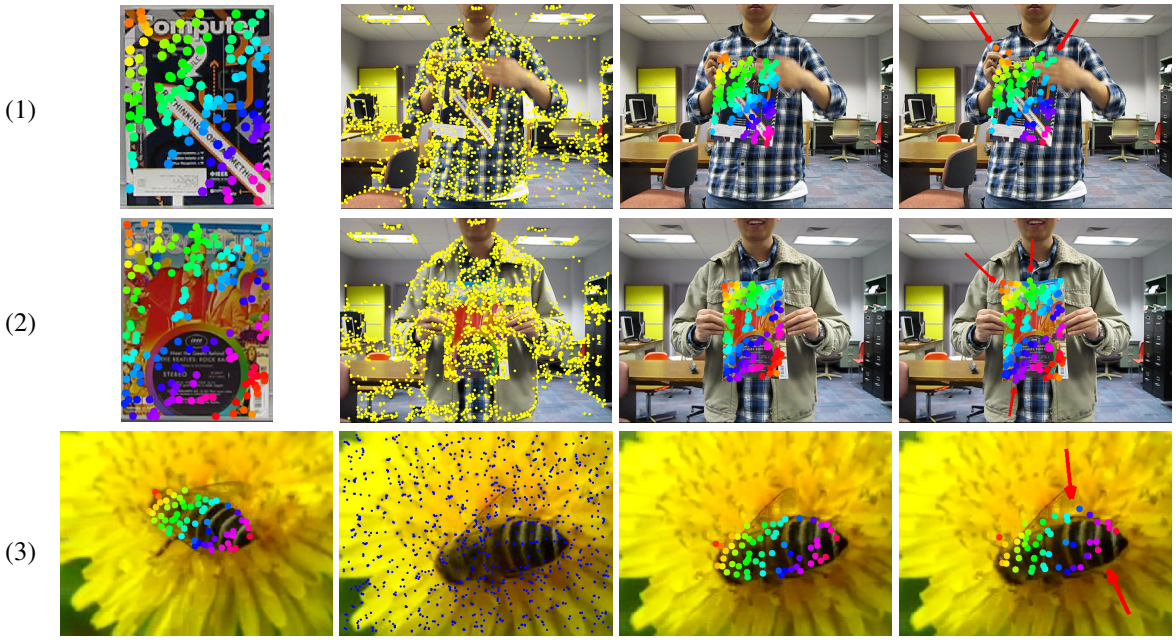
We obtained 3 video clips from the website of the authors of [11] and matched object templates to the instances of objects in the videos frame by frame. Fig. 8.(a) shows the three object templates, and Fig. 8.(c) shows example matching results from the three videos.

For different video clips, we chose the most appropriate optimization model from (11), (13), and (14), which best describes that object's transformation and provides as much geometric constraint as possible. The previous method [11] was used for comparison. The first object (Fig. 8.(1)) is an IEEE Spectrum magazine mainly undergoing global transformation. We chose to model its transformation using the global affine transformation model (11), *i.e.*, $\Theta \in \mathbf{R}^6$. The second object is a Computers 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 (13) to match this object. The third object is a bee working on a flower. This real-world object undergoes complex transformations, and therefore we again used the locally affine model to approximate its transformations. Figs. 8.(c) and 8.(d) show the comparison between our results and the results obtained by [11]. It is obvious




(a) (b) (c) (d) (e)

Figure 7. Our method's matching results on four image pairs obtained from [12]. (a) Template images. (b) Template SIFT feature points. (c) Scene images. (d) Scene SIFT feature points. (e) Matching results in scene images.




(a) (b) (c) (d)

Figure 8. Example matching results from the three videos by our proposed method and the previous method [11]. (a) Template feature points. (b) The scene images. Dots represent all detected feature points. (c) The matching results by our proposed method. (d) The matching results by the previous method [11].

Figure 9. Four example frames' matching results from the toy worm video by our proposed method. Yellow dots on the background represent all detected feature points.

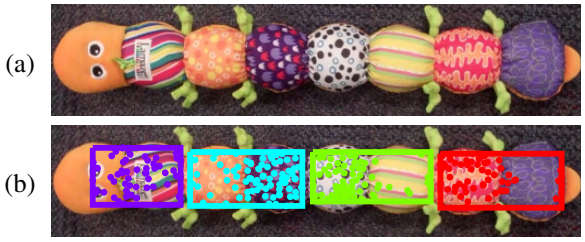



Figure 10. (a) The template image of the toy worm. (b) 4 parts' template feature points are colored differently. Boxes are drawn for better visualization in the matching results.

that our results are much more robust to object occlusion and feature point mis-detection.

### 5.4. Objects Undergoing Articulated Transformation

We also obtain another video from [11], which records a toy worm being bent by a person. We chose to use the articulated transformation model (14) to approximate its complex transformations. We modeled the toy worm as a template consisting of four connected parts undergoing separate affine transformations, *i.e.*, $\Theta_{G_u} \in \mathbf{R}^6, u = 1, \cdots, 4$. Fig. 10.(b) shows template feature points belonging to the four parts. For better visualization, we only show the four parts' bounding boxes in the results. Fig. 9 shows 4 example matching results from the video sequence. Although the toy undergoes very complex transformations, our proposed articulated model (14) was able to match it satisfactorily.

## 6. Conclusion and Discussion

In this paper, we present a novel object matching method based on convex optimization techniques. Convex feature cost functions are created by relaxing the original discrete feature cost functions. Such convex functions result in convex optimization models that allow multiple levels of degrees of freedom on transformation. These models can better constrain transformations of different types of object templates to generate more accurate matching results. Our method's "soft" matching results are also more robust to feature point occlusion or mis-detection. Extensive experiments and comparison with previous methods demonstrate the effectiveness and robustness of our proposed method.

Because of the convex relaxation, the optimal solution obtained in our method does not always correspond to the true optimal matching solution. When the feature points are distinctive, the relaxation using the lower convex hull is tight, whereas the relaxation could bring large errors if feature points are very indistinctive. In real-world applications, feature points usually have distinctive power, thus having large errors due to the relaxation is very rare as demonstrated by extensive experiments.

**Acknowledgments.** This research was supported in part by the NSF grant IIS-0812120. The authors would like to thank Dr. Chenyang Xu (Siemens) for stimulating discussions on the transformation models.

## References

[1] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. *Proc. CVPR*, pages 26–33, 2005.

[2] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE Trans. PAMI*, 10:2349–2374, 2009.

[3] S. Choi, T. Kim, and W. Yu. Performance evaluation of ransac family. *Proc. BMVC*, 2009.

[4] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. *Proc. NIPS*, pages 313–320, 2006.

[5] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *Proc. CVPR*, 2009.

[6] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. `http://cvxr.com/cvx`, Feb. 2011.

[7] H. Jiang, M. S. Drew, and Z. Li. Matching by linear programming and successive convexification. *IEEE Trans. PAMI*, 29:959–975, 2007.

[8] H. Jiang and S. X. Yu. Linear solution to scale and rotation invariant object matching. *Proc. CVPR*, 2009.

[9] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. *Proc. ICCV*, pages 1482–1489, 2005.

[10] M. Leordeanu and M. Hebert. Unsupervised learning for graph matching. *Proc. CVPR*, 2009.

[11] H. Li, E. Kim, X. Huang, and L. He. Object matching with a locally affine-invariant constraint. *Proc. CVPR*, 2010.

[12] H. Ling and D. W. Jacobs. Deformation invariant image matching. *Proc. ICCV*, 2005.

[13] H. Liu and S. Yan. Common visual pattern discovery via spatially coherent correspondences. *Proc. CVPR*, 2010.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Comp. Vis.*, 60:91–110, 2004.

[15] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. PAMI*, 19:530–535, 1997.

[16] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. *Proc. CVPR*, 2008.