Object Matching Using A Locally Affine Invariant and Linear Programming Techniques

Hongsheng Li, Xiaolei Huang, *Member, IEEE*, and Lei He,

Hongsheng Li and Xiaolei Huang are with the Department of Computer Science and Engineering at Lehigh University, Bethlehem, PA, USA. Lei He is with Library of Congress, Washington DC, USA.

Abstract

In this paper, we introduce a new matching method based on a novel locally affine-invariant geometric constraint and linear programming techniques. To model and solve the matching problem in a linear programming formulation, all geometric constraints should be able to be exactly or approximately reformulated into a linear form. This is a major difficulty for this kind of matching algorithms. We propose a novel locally affine-invariant constraint which can be exactly linearized and requires a lot fewer auxiliary variables than other linear programming based methods do. The key idea behind it is that each point in the template point set can be exactly represented by an affine combination of its neighboring points, whose weights can be solved easily by least squares. Errors of reconstructing each matched point using such weights are used to penalize the disagreement of geometric relationships between the template points and the matched points. The resulting overall objective function can be solved efficiently by linear programming techniques. Our experimental results on both rigid and non-rigid object matching show the effectiveness of the proposed algorithm.

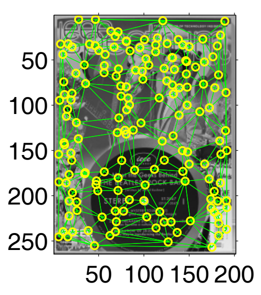
Index Terms

Feature matching, object matching, locally affine invariant, linear programming

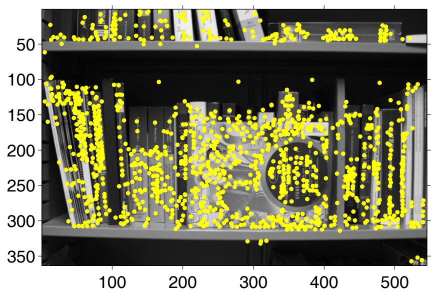
I. Introduction

The problem of object matching in 2D images can be defined as matching a group of *template* feature points representing an object to an instance of that object in a given *scene* image (Fig. 1). Each feature point has a location (x, y) in the 2D image domain and a feature vector describing the object's local appearance around that location. Matched scene feature points should maintain consistency with the template points in both local appearance and relative spatial (neighborhood) relationships. The lines in Fig. 1 represent neighborhood relationships between template feature points, i.e., if two feature points are connected by a line, they are neighbors. Object matching has extensive uses in object recognition [6], detection and tracking [21], shape matching [21], and image retrieval [36].

Assuming a global transformation between two groups of feature points, Random Sample Consensus (RANSAC) [17] and its large number of variants [32] have been widely used to solve matching problems. Although the RANSAC has the ability to tolerate a tremendous fraction of outliers, such an ability is obtained by assuming only a global transformation between the two



(a)



(b)

Fig. 1

(A) A GROUP OF *template* FEATURE POINTS (YELLOW CIRCLES) REPRESENTING A SPECTRUM MAGAZINE. GREEN LINES SPECIFY NEIGHBORHOOD RELATIONSHIPS BETWEEN POINTS. (B) THOUSANDS OF FEATURE POINTS IN AN INPUT *scene* IMAGE.

groups of feature points. Therefore, the RANSAC methods have difficulties handling objects undergoing complex transformations.

The object matching problem has also been extensively studied as a graph matching problem [15], [18]. Leordeanu and Hebert [23] proposed a spectral method working on a matrix where the diagonal elements represent one-to-one assignment costs, and other elements represent pairwise agreements between potential correspondences. The correspondences are then obtained by finding the principal eigenvector of this matrix. This method uses distances between pairs of points as the geometric constraints, which are only rotationally invariant. Cour et al. [14] proposed a spectral relaxation method for the graph matching problem that incorporates one-to-one or one-to-many mapping constraints, and presented proper bistochastic normalization of the graph matching compatibility matrix to improve the overall matching performance. Other spectral methods for graph matching includes [8], [11], [37], [41]. Torresani et al. [39] 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. Torki and Elgammal [38] formulated the consistent matching problem as an embedding problem where the goal is to embed all the feature matching costs and spatial arrangments in a Euclidean space. Correspondences are then recovered by a bipartite matching on the embedded points. Liu and Yan [26] proposed an algorithm to discover all common visual patterns within two sets of feature points. It optimizes the same objective function as that of [23] 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. Other graph matching methods can be categorized as relaxation labeling and probabilistic approaches [9], [12], [22], [25], [33], [42], semidefinite relaxation [35], replicator equations [31], tree search [29], and RKHS methods [40]. To automatically set the weights of different terms in the similarity matrix, supervised [10] and unsupervised [24] learning methods optimizing the weights were proposed. However, one major limitation of the graph matching methods is that order-2 edges can only provide at most rotational invariant. Zass and Shashua [43] extended ordinary graphs to hypergraphs, whose high-order edges can encode more complex geometric invariants. The method's output is a probabilistic ("soft") result rather than traditional "hard" node-to-node results. In this way, they were able to model the problem as a convex optimization problem and obtained a global minimum. Duchenne et al. [16] used high-order (mostly 3 or 4) constraints instead of unary or pairwise ones between template points, which result in a tensor representing affinity between feature tuples. The resulting energy function can then be optimized using the power iteration method.

The matching problem has also been modeled as mathematical programming problems. Chui and Rangarajan [13] interpreted it as a mixed variable (binary and continuous) optimization problem. The correspondence problem is viewed as a linear assignment solved by softassign and deterministic annealing. Berg et al. [6] modeled the matching problem as a quadratic integer programming problem. It uses pairwise relationships between feature points and penalizes both rotation and scaling differences. Recently, linear programming has been used in object matching. Jiang et al. [20] proposed a linear solution to the feature matching problem. The main difficulty of this framework is to find geometric constraints which can be exactly or approximately linearized. In [20], the vectors defined by pairwise points are used as the geometric constraints for its objective function. They can only tolerate small local deformations and is not invariant to global transformations, such as similarity or affine transformations. To solve this problem, Jiang and Yu [21] explicitly modeled scaling and rotation, and approximated the resulting formulation by a convex program. The resulting solution is invariant to global rotation and scaling. Its extensive experimental results demonstrated the effectiveness and robustness of the pairwise geometric constraint in various object matching scenarios. In [19], a linear augmented tree model was proposed, which allows arbitrary metrics for the pairwise costs on trees and it also allows high-order constraints that couple all the nodes. After linearization, its objective function can be efficiently optimized by dynamic programming. Zheng et al. [44] jointly optimized the correspondences and transformations between feature points. The final objective function is alternatively optimized in two steps. In one step, correspondences and transformations is optimized by a linear programming model, but the resulting correspondences are continuous; in another step, the continuous correspondences are mapped to the discrete solution space.

Along this line, we propose a new locally affine-invariant geometric constraint for the linear programming based matching framework. For each template point, we represent it as an affine combination of its neighboring points. Such affine combinations can be easily and efficiently solved by least squares. As demonstrated in the next section, these representations are invariant to affine transformations. Moreover, since the coefficients of each affine combination are calculated by using only its corresponding point's neighboring points, this constraint is a local one.

Our new geometric constraint has three major advantages: (i) our proposed geometric constraint is locally affine-invariant. Therefore, it can handle more complex and natural transformations of an object. (ii) Unlike the approximate linearization of the similarity-invariant constraint in [21], the exact linearization of our new constraint requires much fewer auxiliary variables. Therefore, it is asymptotically faster and is also easier to implement. (iii) For each template point, all of its neighboring points are used to calculate the affine combination coefficients. It is a higher order geometric constraint, which is more distinctive and can better exclude ambiguous matchings [16].

II. Methodology

A. Problem Formulation

Let n_t and n_s be the numbers of template and scene feature points respectively, $T \in \mathbf{R}^{n_t \times 2}$ and $S \in \mathbf{R}^{n_s \times 2}$ be the matrices recording template points' and scene points' 2D coordinates respectively, $\mathbf{p}_i = [x_i, y_i]^T \in \mathbf{R}^2$ and $\mathbf{q}_j \in \mathbf{R}^2$ be the ith template and the jth scene points' coordinates, and $\mathcal{N}_{\mathbf{p}_i}$ be the set of ordered points in the neighborhood of $\mathbf{p}_i$. The order of points in each neighborhood is randomly set. The matching function $m(\cdot)$ matches every template feature point $\mathbf{p}_i$ to a feature point $m(\mathbf{p}_i)$ in the scene set. The goal is to find the matching function $m(\cdot)$

that minimizes the overall objective function consisting of both feature and geometric matching costs:

$$\sum_{i=1}^{n_t} \left\{ c(\mathbf{p}_i, m(\mathbf{p}_i)) + \lambda \cdot g(\mathbf{p}_i, \mathcal{N}_{\mathbf{p}_i}; m(\mathbf{p}_i), \mathcal{N}_{m(\mathbf{p}_i)}) \right\}, \tag{1}$$

where $c(\mathbf{a}, \mathbf{b})$ is the feature matching cost between the feature points $\mathbf{a}$ and $\mathbf{b}$, $g(\cdot)$ is the geometric cost function that measures the geometric dissimilarity between two sets of ordered points $\{\mathbf{p}_i, \mathcal{N}_{\mathbf{p}_i}\}$ and $\{m(\mathbf{p}_i), \mathcal{N}_{m(\mathbf{p}_i)}\}$, and λ controls the relative weight between the feature and geometric cost terms.

The choice of features is not restricted to similarity or affine invariant ones, e.g., SIFT [27]. For general non-transformation-invariant features, the matching cost between two feature points $\mathbf{a}$ and $\mathbf{b}$, $c(\mathbf{a}, \mathbf{b})$, can be defined by the minimal distance across all possible similarity or affine transformations T with parameters Θ,

$$c(\mathbf{a}, \mathbf{b}) = \min_{\Theta} distance(feature(\mathbf{a}), feature(T(\mathbf{b}; \Theta))). \tag{2}$$

The feature matching costs between every template point and every scene point are pre-calculated before the matching is performed. They are stored in a feature matching cost matrix $C \in \mathbf{R}^{n_t \times n_s}$, where C_{ij} stores the cost of matching the ith template feature point to the jth scene feature point.

For the geometric constraints, unlike the formulation proposed in [20], [21], where only pairwise geometric relationships are considered, our new formulation takes into consideration of higher order (at least order 3) geometric constraints, which are more distinctive and therefore can better exclude ambiguous matchings [16]. The neighborhood $\mathcal{N}_{\mathbf{p}_i}$ of $\mathbf{p}_i$ is pre-defined before the matching is performed. It remains an open issue how to properly define neighboring relationships $\mathcal{N}_{\mathbf{p}_i}$ for each template point $\mathbf{p}_i$ to better representing different objects. In this paper, we tested two approaches: Delaunay Triangulation and k-nearest-neighbor (kNN). Detailed discussion of the two approaches is in Section II-H. In the next two subsections, we first present a way to model the matching function $m(\cdot)$ in (1) and then introduce a novel locally affine-invariant geometric constraint for the geometric cost function $g(\cdot)$.

B. The Modeling of the Feature Matching Function

The matching function $m(\cdot)$ is usually modeled as a set of binary variables [14], [21], [23]. Similarly, we define a binary variable matrix $X \in \{0, 1\}^{n_t \times n_s}$ to represent the matching function

$m(\cdot)$. $X_{ij} = 1$ or 0 denotes the matching between the ith template feature point and the jth scene feature point is either “Yes” or “No”. Each row of X contains exactly one 1, meaning every template point must be matched to exactly one point in the scene image.

We can then represent the first term in (1), the feature matching cost term, as

$$tr(C^T X) = \sum_{i}^{n_t} \sum_{j}^{n_s} C_{ij} X_{ij}. \tag{3}$$

Because there is only one 1 in each row of X, only one feature cost for each template point (in each row) of C would be added into the feature cost term.

For the ith row of X, the column index of the 1 in this row, specifies which scene point $\mathbf{p}_i$ would choose as its corresponding scene point. Let X_i denote the ith row of X, $X_i S$ calculates the matched scene point's coordinates for $\mathbf{p}_i$. Combining all rows of X, XS calculates the matched scene feature points' coordinates in the same order as the template points.

C. A Locally Affine-Invariant Constraint

The major difficulty of modeling (1) is to define the geometric cost function $g(\cdot)$. We propose a geometric constraint to implicitly model it. Our geometric constraint has two requirements on each template point's neighborhood: (i) every template point must have at least three neighbors, and (ii) every template point's neighboring points must not be collinear, i.e., they do not lie on a single straight line. Our goal is to create a way to characterize the geometric properties of the neighborhood of each template point. To do so, we assume each $\mathbf{p}_i$ can be exactly represented by an affine combination of its neighboring points, i.e.,

$$\mathbf{p}_i = \sum_{\mathbf{p}_j \in \mathcal{N}_{\mathbf{p}_i}} W_{ij} \mathbf{p}_j, \tag{4}$$

where W is a $n_t \times n_t$ weight matrix recording the affine combination coefficients for all template points, and W_i is the ith row of W recording the affine combination coefficients for $\mathbf{p}_i$. Intuitively, W_i reveals the local geometric layout around $\mathbf{p}_i$. There are two constraints on the weight matrix W: $W_{ij} = 0$ if $\mathbf{p}_j \notin \mathcal{N}_{\mathbf{p}_i}$, and each row must sum to one (equivalently, each point is represented by an affine combination of its neighbors). The first constraint reflects that this matrix only describes the local geometric properties of each point. The second makes the representation invariant to global translation.

It is easy to prove that a point can always be exactly represented by the affine combination of its neighbors, if the above mentioned two requirements are satisfied. Assuming $\mathbf{p}_i$ has only three neighbors $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$, the affine combination coefficients W_i for $\mathbf{p}_i$ can be obtained by first solving the following linear equations:

$$\begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 \\ 1 & 1 & 1 \end{bmatrix} \tilde{W}_i^T = Q\tilde{W}_i^T = \begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix}. \tag{5}$$

Because $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ are not collinear, the matrix Q has full rank. $\tilde{W}_i^T = Q^{-1}[\mathbf{p}_i^T \ 1]^T$ is the exact solution of the affine combination coefficients for $\mathbf{p}_i$. We can then fill W_i using $\tilde{W}_i$: $W_{ij} = \tilde{W}_{il}$ if $\mathbf{p}_j$ is the lth neighbor of $\mathbf{p}_i$, and $W_{ij} = 0$ if $\mathbf{p}_j \notin \mathcal{N}_{\mathbf{p}_i}$. If $\mathbf{p}_i$ has more than 3 neighbors, we can still obtain an exact affine combination by just using the first three neighbors and setting all other neighbors' weights to 0. In practice, we use least squares to minimize the error of each point's affine combination. Since least squares guarantees obtaining a solution with minimal error under $L2$ norm, and we just showed at least one solution with zero error exists, the solution by least squares is also an exact representation of that point. Although there might be an infinite number of affine representations for a point, any one of them can be used in our framework. We choose least squares because one of its desired properties is that it tends to assign nonzero weights to all neighbors [7], which means that the local geometric properties of each point are described by all of its neighbors.

We calculate the reconstruction weights $\tilde{W}_i$ for each point $\mathbf{p}_i$ separately and transform them into the matrix form W by the aforementioned scheme. The representation error for any template point $\mathbf{p}_i$ is always zero no matter what type of norm is used, i.e.,

$$\left\| \mathbf{p}_i - \sum_j W_{ij}\mathbf{p}_j \right\|_{0,1,2,\cdots,F} = 0, \qquad \text{for } i = 1, \cdots, n_t. \tag{6}$$

For this particular method, we choose $L1$ norm for the representation error, since it can be exactly linearized (Section II-F). Obviously, the error function (6) is affine invariant:

$$\begin{aligned} 0 &= \left\| \mathbf{p}_i - \sum_j W_{ij}\mathbf{p}_j \right\|_1 \\ &= \left\| A\mathbf{p}_i - \sum_j W_{ij}A\mathbf{p}_j \right\|_1 \end{aligned} \tag{7}$$

$$= \left\| (\mathbf{p}_i + t) - \sum_j W_{ij} (\mathbf{p}_j + t) \right\|_1 ,$$

where A and t denote an arbitrary 2×2 affine transformation matrix and an arbitrary 2×1 translation vector, respectively. Summing up all template points' reconstructions (7) and reformulating them into a matrix form result in

$$\|(I - W)T\| = 0, \tag{8}$$

where $I \in \mathbf{R}^{n_t \times n_t}$ is the identity matrix, $T \in \mathbf{R}^{n_t \times 2}$ records template points' 2D coordinates, and $|| \cdot ||$ denotes the summation of all absolute values of elements in a matrix.

As we mentioned in Section II-B, XS represents the matched scene feature points in template points' order. Therefore, substituting XS for T in (8) leads to our geometric cost term, the second term in (1). Without any feature information, we seek the $X \in \mathbf{R}^{n_t \times n_s}$ matrix which best preserves the geometric properties of the template point set specified by its weight matrix W:

$$\arg\min_X \ \|(I - W)XS\| . \tag{9}$$

However, there are degenerate cases. On one hand, matching all template points to one scene point also leads to a zero geometric cost because $\sum_j W_{ij} = 1$. Fortunately, in the object matching tasks, features have discriminative power. Those degenerate cases usually result in very large feature costs and thus are not likely to be the optima of the objection function (1). Even when the features used are not distinctive enough, we can further add constraints into our optimization model to explicitly exclude those degenerate cases (Section II-E). On the other hand, some parts of an object may be folded. If the features are invariant to such local deformations, matching several template points to one scene point also minimizes the error function (6) and should be considered as a correct matching (Section III-G).

D. Relation to Locally Linear Embedding [34]

Our affine invariant is inspired by the Locally Linear Embedding (LLE) and has a similar formulation, but our invariant is different from LLE in essence. Our invariant assumes each point being represented by an "affine" combination of its neighboring points, while LLE assumes a "convex" combination. Reconstruction errors by our proposed invariant are affine-invariant (7). In contrast, LLE's reconstruction error for each point is not transformation-invariant, thus its "convex" combination cannot be used in this matching framework.

E. The Overall Objective Function

Summing up the feature cost term (3) and the geometric cost term (8) leads to our overall objective function:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & tr(C^T X) + \lambda \left\| (I - W) X S \right\| \\ \text{subject to} \quad & X \in \{0,1\}^{n_t \times n_s}, \\ & X \mathbf{1}_{n_s} = \mathbf{1}_{n_t}, \\ & X^T \mathbf{1}_{n_t} \leq \mathbf{w}_{n_s} \text{ (optional)}, \end{aligned} \tag{10}$$

where $\mathbf{w}_{n_s} \in \mathbf{R}^{n_s}$ denotes a column vector of n_s constant number ws.

There are three constraints:

- $X \in \{0,1\}^{n_t \times n_s}$ denotes the matching between a template and a scene feature point is either “Yes” (1) or “No” (0).
- $X \mathbf{1}_{n_s} = \mathbf{1}_{n_t}$ denotes all template points should be matched into the scene point set. If one template point’s corresponding scene point is occluded or not detected, minimization of the objective function would prefer matching it to another scene point which well approximates that template point’s local geometric properties.
- $X^T \mathbf{1}_{n_t} \leq \mathbf{w}_{n_s}$ allows matching at most w ($w < n_t$) template points to one scene point and thus avoids the degenerate cases we mentioned in Section II-C. However, in practice, this constraint is usually not necessary since matching all template points to one scene point usually leads to a very large feature matching cost. We used this constraint in Section III-A, III-B ($w = 1$) and Fig. 13.(4) ($w = 4$).

F. Linearization and Relaxation

The problem (10) has a nonlinear objective function with integer constraints. It is NP-hard and cannot be efficiently solved. However, because $\lambda > 0$, the second term of (10) can be exactly linearized in the following way:

$$\begin{aligned} \underset{x_i}{\text{minimize}} \quad \textstyle\sum_{i=1}^{N} |x_i| \Leftrightarrow \underset{x_i,\, u_i}{\text{minimize}} \quad & \textstyle\sum_{i=1}^{N} u_i \\ \text{subject to} \quad & x_i \leq u_i,\ x_i \geq -u_i \\ & \text{for all } i = 1, \cdots, N, \end{aligned}$$

where u_i is the ith auxiliary variable representing the upper bound of $|x_i|$.

We further relax the binary constraints, $X \in \{0,1\}^{n_t \times n_s}$, to a continuous domain $[0,1]^{n_t \times n_s}$ to convert the original problem (10) into a linear programming (LP) form:

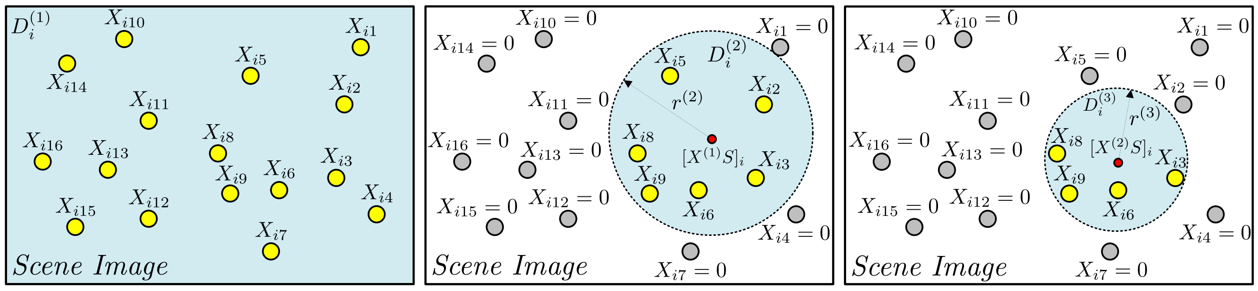
$$\begin{aligned} \text{minimize} \quad & f(X) = tr(C^T X) + \lambda \mathbf{1}_{n_t}^T U \mathbf{1}_2 && (11) \\ \text{subject to} \quad & X \geq 0, \\ & X\mathbf{1}_{n_s} = \mathbf{1}_{n_t}, \\ & (I-W)XS \leq U, \\ & (I-W)XS \geq -U, \\ & X^T \mathbf{1}_{n_t} \leq \mathbf{w}_{n_s} \, \text{(optional)}, && (12) \end{aligned}$$

where $U \in \mathbf{R}^{n_t \times 2}$ is an auxiliary variable matrix representing upperbounds for each entry of $\|(I-W)XS\|$.

G. Numerical Scheme

Without any simplification trick, the number of variables in our LP model (11) is proportional to $n_t \times n_s$. In contrast, the number of variables of the LP model in [21] is proportional to $n_t \times n_s \times$ *the number of scaling discretizations*. Moreover, in the first step of the LP method in [21], it needs to solve 4 such LP problems because it models rotation as 4 different linear constraints. Therefore, our algorithm is asymptotically faster than that in [21].

We utilize the successive trust region shrinkage method proposed in [20] to solve our LP problem (11). For each template point $\mathbf{p}_i$, we set a trust region D_i in the scene image, only scene points inside its trust region are considered as the template point's matching candidates. For instance, for a template point $\mathbf{p}_i$, if $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$ are inside and $\mathbf{q}_4$ is outside its trust region. Then only X_{i1}, X_{i2}, and X_{i3} need to be optimized in (11), and X_{i4} is fixed to 0 during the optimization process. We successively shrink each template point's trust region and refine its matching candidates to gradually obtain accurate matching results. In the first iteration, for each template point $\mathbf{p}_i$, we set its trust region $D_i^{(1)}$ as the entire scene image, and all scene feature points are used in the optimization model (11) (Fig. 2.(a)). The continuous result in the domain $[0,1]^{n_t \times n_s}$ obtained in the first iteration is denoted as $X^{(1)}$, and the resulting matched scene points can be calculated as $X^{(1)}S$. We denote the ith row of $X^{(1)}S$ as $[X^{(1)}S]_i$, which are the



(a) (b) (c)

Fig. 2

ILLUSTRATION OF THE SUCCESSIVE TRUST REGION SHRINKAGE SCHEME IN SECTION II-G FROM A TEMPLATE POINT $\mathbf{p}_i$'S VIEW. (A) IN THE FIRST ITERATION, ALL SCENE POINTS ARE CHOSEN AS MATCHING CANDIDATES FOR THE TEMPLATE POINT $\mathbf{p}_i$. (B) IN THE SECOND ITERATION, ONLY SCENE POINTS INSIDE THE TRUST REGION $D_i^{(2)}$ ARE CHOSEN AS MATCHING CANDIDATES FOR $\mathbf{p}_i$. BINARY VARIABLES CORRESPONDING TO MATCHING TO OTHER SCENE POINTS ARE FIXED TO 0. (C) IN THE THIRD ITERATION, ONLY SCENE POINTS INSIDE THE TRUST REGION $D_i^{(3)}$ ARE CHOSEN AS MATCHING CANDIDATES FOR $\mathbf{p}_i$. BINARY VARIABLES CORRESPONDING TO MATCHING TO OTHER SCENE POINTS ARE FIXED TO 0.

coordinates of the ith matched scene point. In the second iteration, for each template point $\mathbf{p}_i$, we set a trust region $D_i^{(2)}$ with diameter $r^{(2)}$ centered at $[X^{(1)}S]_i$ such that not all scene points would be inside its trust region. Only scene points inside each trust region are then considered as matching candidates for that template point; other scene points are ignored for this template point, i.e., binary variables corresponding to matching to them are always set to zero (Fig. 2.(b)). The resulting matched scene points' coordinates obtained in the second iteration are calculated as $X^{(2)}S$. In the third iteration, a smaller $r^{(3)}$ is set so each template point has fewer matching candidates in the trust region $D_i^{(3)}$ (Fig. 2.(c)). Similar operations are then performed in latter iterations. To map the final continuous results obtained in the nth iteration, $X^{(n)}$, to the discrete solution space, we fix all but one rows of $X^{(n)}$, and try to set 1 to each column of the row that are not fixed. The column with the minimum objective function value is then set to 1 for that row. We perform this operation for all rows. In this way, we obtain a discrete X with exactly one 1 in each row.

The above scheme works efficiently when both the numbers of template points and scene points are small (less than 100). When the numbers of features points are large, i.e., the size of

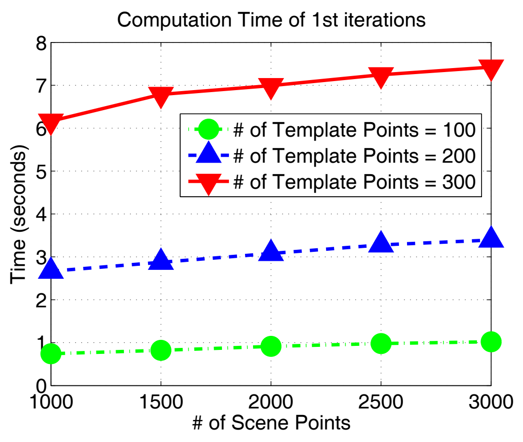


Fig. 3

ACTUAL COMPUTATION TIME OF 1ST ITERATIONS OF OUR PROPOSED METHOD WITH DIFFERENT NUMBERS OF TEMPLATE AND SCENE FEATURE POINTS.

X is very large. We use the lower convex hull trick in [20] to reduce the computation complexity. For each template point, we view its matching scene candidates as a 3D point cloud with the 3rd dimension as their feature costs. A lower convex hull with respect to the 3rd dimension is calculated and only scene candidates on the convex hull are further refined as the matching scene candidates. In this way, the number of each template point's matching candidates is further reduced and is usually less than 100.

LP with tens of thousands of variables and thousands of constraints can be solved within seconds on a standard PC using state-of-the-art solvers, such as CPLEX and Gurobi. In our experiments, we use MATLAB with a non-commercial solver, *lpsolve* [1], which employs the simplex methods. Typically, to match 100 template points and thousands of scene points, each LP iteration takes less than 1.5 second on an Intel E6850 3.0GHz CPU, and the LP trust region shrinkage runs for 4-8 iterations. Note that the running time can be further shortened by implementing the method in C/C++.

We performed an empirical speed test using synthetic data to test our method's computation time with varying numbers of template and scene points. The template and scene points' coordinates and the feature cost matrix were randomly generated. We increased the number of template points from 100 to 300 and the number of scene points from 1000 to 3000. Every

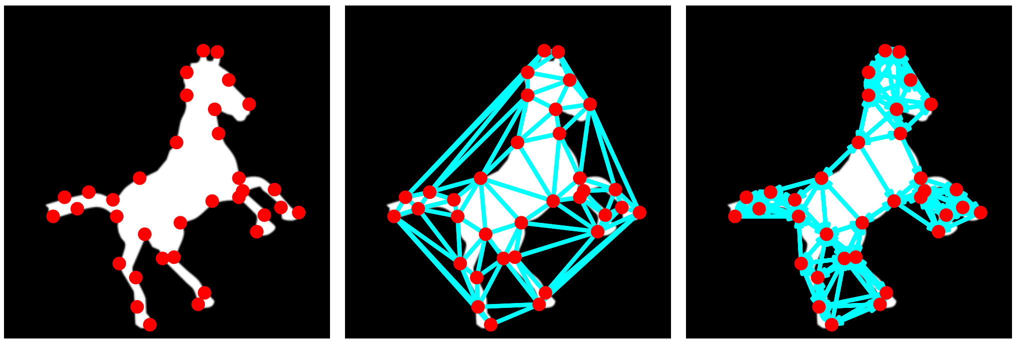

(a) (b) (c)

Fig. 4

ILLUSTRATION OF NEIGHBORHOODS DEFINED BY DELAUNAY TRIANGULATION (DT) AND kNN. (A) A TEMPLATE POINT SET ON THE SILHOUETTE OF A HORSE SHAPE [4]. (B) NEIGHBORHOODS DEFINED BY DELAUNAY TRIANGULATION. IF TWO TEMPLATE POINTS ARE CONNECTED BY A LINE, THEY ARE NEIGHBORS. (C) NEIGHBORHOODS DEFINED BY kNN WITH $k = 5$. IF AN ARROW IS POINTED FROM POINT A TO POINT B, B IS A'S NEIGHBOR.

combination of numbers of points was tested and the computation time of the 1st iteration with most scene points was recorded (Fig. 3). The computation time increases sub-linearly as the number of scene points increases and quadratically as the number of template points increases. Therefore, the computation time depends mainly on the number of template points but much less on the number of scene points.

H. Two Ways of Defining Neighborhoods

It remains an open issue how to define meaningful neighborhoods for template points in different applications. In this paper, we tested two ways of defining neighborhoods: Delaunay Triangulation (DT) and kNN with $k = 5$, 9, or 13. As we observed in our experiments (Section III-A, III-B, III-C and III-D), the two ways result in similar matching performance. However, in some cases, one of the approaches might generate smaller matching errors. Generally speaking, DT is more suitable for matching an object transforming globally (e.g., matching cases in Section III-B) while kNN better tolerates objects' local deformations (e.g., matching cases in Section III-A). Note that in some cases Delaunay Triangulation might associate a template point with only two neighbors. In such cases, we randomly choose another point near it as its 3rd neighbor

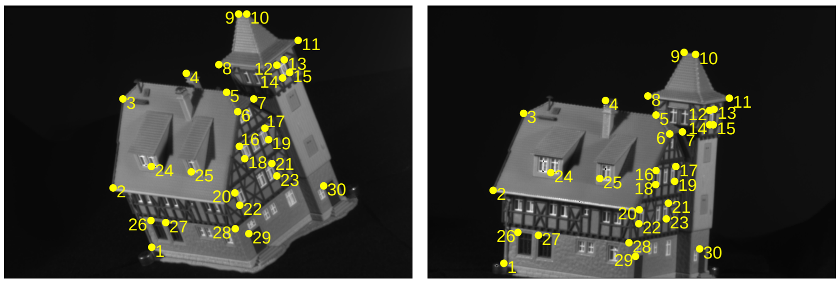


Fig. 5

AN EXAMPLE MATCHING CASE FROM THE *house* SEQUENCE WITH FRAME SEPARATION LEVEL = 90. (LEFT) THE 1ST FRAME AND MANUALLY LABELED LANDMARKS IN IT, WHICH ARE USED AS TEMPLATE POINTS. (RIGHT) THE LABELED POINTS IN THE 91ST FRAME ARE USED AS SCENE FEATURE POINTS. OUR METHOD'S MATCHING RESULTS ARE LABELED BY NUMBERS IN THE FIGURE. A 0.0% MATCHING ERROR IS ACHIEVED IN THIS CASE.

to fulfill the three-neighbor requirement mentioned in Section II-C. We illustrate the differences between the two ways using one example in Fig. 4. In Fig. 4.(a), points on the silhouette of a horse shape [4] are used as template points. Neighborhoods defined by Delaunay Triangulation are shown as lines in Fig. 4.(b). Points on the convex hull are defined as neighbors although they might be far away from each other. For instance, points on the head and the tail are defined as neighbors; therefore, the geometric cost term would penalize the difference between the head's and the tail's transformations. In contrast, as shown in Fig. 4.(c), neighbors defined by kNN are more locally connected. The geometric cost term would less penalize the difference between transformation between the head and the tail.

III. EXPERIMENTS

In our experiments, we used Shape Context [5] with its default parameter setting as features for the first three experiments and SIFT features [27] for the remaining ones. Feature matching cost is calculated as the $L2$ distance between two feature vectors. For each matching case, we normalized feature matching costs with respect to their maximum value to let them span the range $[0, 1]$, and set $\lambda = 0.05$, 1 or 10 depending on how flexible the object is. For the first four experiments, we tested two ways of defining neighborhoods: Delaunay Triangulation (DT) and kNN with $k = 5$, 9 and 13. For the rest of the experiments, we used DT. We measured

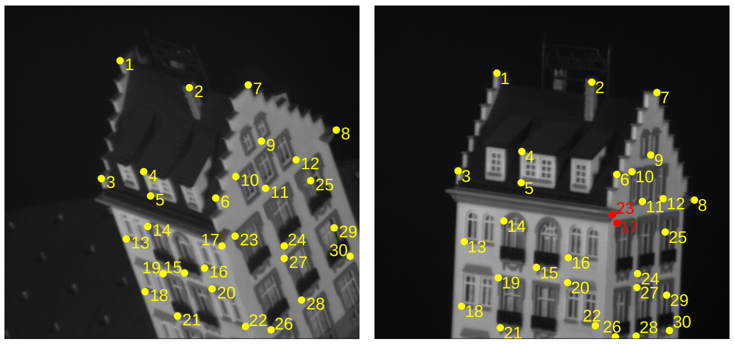


Fig. 6

AN EXAMPLE MATCHING CASE FROM THE *hotel* SEQUENCE WITH FRAME SEPARATION LEVEL = 90. (LEFT) THE 2ND FRAME AND MANUALLY LABELED LANDMARKS IN IT, WHICH ARE USED AS TEMPLATE POINTS. (RIGHT) THE LABELED POINTS IN THE 92ND FRAME ARE USED AS THE SCENE FEATURE POINTS. OUR METHOD'S MATCHING RESULTS ARE LABELED BY NUMBERS IN THE FIGURE. 2 OUT OF 30 TEMPLATE POINTS ARE WRONGLY MATCHED IN THIS CASE (SHOWN IN RED).

different methods' matching errors as either the percentage or the number of wrong matchings. We utilized the lower convex hull trick mentioned in Section II-G for all but the first three experiments. (The first three experiments have a small number of template and scene points, thus requiring no low convex hull speed-up.)

A. *CMU House and Hotel Sequences [3], [2]*

In our first experiment, we used the CMU *House* [3] and *Hotel* [2] sequences to test our method's performance and compare it with those of other methods. The two sequences consist of 111 frames and 101 frames, respectively. We followed the experimental setup in [10]. Each frame is manually labeled with the same 30 landmarks across entire sequences[1]. We evaluated our method's performance by creating image pairs using two frames in a same sequence but are separated by a specific number of in-between frames. All such image pairs are tested as the frame separation level increases from 10 to 90. w was set to 1 in this experiment because we were looking for exact one-to-one matchings. We also set $\lambda = 0.05$ to allow more local deformations. Matching errors were then calculated as the percentage of wrong matchings. We

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

TABLE I

MATCHING ERRORS BY DIFFERENT METHODS WITH VARYING FRAME SEPARATION LEVELS FOR THE CMU *house*, CMU *hotel*, HORSE ROTATION, AND HORSE SHEAR SEQUENCES. RESULTS OF [14] AND [10] ARE OBTAINED FROM [10]. OUR METHOD WITH kNN DEFINED NEIGHBORHOODS GENERATES 0.0% AND 1.04% AVERAGE MATCHING ERRORS ON THE *house* AND THE *hotel* SEQUENCES, RESPECTIVELY. OUR METHOD WITH DELAUNAY TRIANGULATION DEFINED NEIGHBORHOODS GENERATES 0.0% AVERAGE MATCHING ERRORS ON HORSE ROTATION AND HORSE SHEAR SEQUENCES.

Sequence	Methods	Frame Separation (Frames)								
		10	20	30	40	50	60	70	80	90
CMU *house*	Our Method + kNN ($k=5$)	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
	Our Method + DT	0.16%	0.0%	0.22%	0.92%	1.31%	5.53%	13.4%	10.2%	12.1%
	Learning + Linear [10]	0.0%	0.0%	0.0%	0.0%	3.17%	3.92%	7.18%	11.0%	13.3%
	Learning + Quadratic [10]	0.0%	0.10%	0.44%	1.85%	1.01%	17.2%	7.06%	11.5%	13.3%
	Balanced [14]	0.0%	0.0%	0.0%	0.87%	0.0%	1.57%	10.8%	15.7%	22.9%
CMU *hotel*	Our Method + kNN ($k=5$)	0.15%	0.49%	1.03%	1.64%	1.63%	**1.14%**	**2.15%**	**1.59%**	**1.82%**
	Our Method + DT	0.07%	0.16%	0.94%	1.97%	2.22%	1.46%	2.37%	2.22%	4.24%
	Learning + Linear [10]	0.22%	0.24%	2.20%	1.67%	6.47%	12.6%	15.3%	18.6%	25.6%
	Learning + Quadratic [10]	0.56%	1.36%	3.19%	5.0%	7.64%	12.3%	12.7%	16.7%	7.78%
	Balanced [14]	**0.0%**	**0.0%**	**0.0%**	**1.0%**	**1.37%**	12.1%	25.7%	28.6%	34.4%
Horse Rotation	Our Method + kNN ($k=5$)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	21.2%
	Our Method + DT	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
	Learning + Linear [10]	0.0%	2.19%	15.5%	51.6%	83.0%	89.0%	91.2%	87.3%	87.0%
	Learning + Quadratic [10]	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.84%	0.08%
	Balanced [14]	0.0%	0.0%	0.0%	0.0%	3.65%	25.5%	52.1%	68.2%	67.2%
Horse Shear	Our Method + kNN ($k=5$)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	Our Method + DT	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
	Learning + Linear [10]	0.10%	0.16%	0.62%	0.63%	1.21%	4.63%	4.72%	7.39%	9.68%
	Learning + Quadratic [10]	0.52%	1.42%	2.73%	7.43%	15.5%	20.2%	23.8%	34.7%	37.1%
	Balanced [14]	0.0%	0.0%	0.0%	0.0%	4.69%	5.72%	13.5%	26.6%	38.0%

tested our method with neighborhoods defined by Delaunay Triangulation and kNN. The results show that the kNN with $k = 5$ generates the best matching performance. This is due to the existence of large local deformations between template and scene points. We compared our method with the balanced graph matching [14] and the learning-based graph matching method [10] with linear and quadratic objective functions.

The first row of Table I shows the matching errors obtained by the tested methods on the *house* sequence. Our method with kNN ($k = 5$) defined neighborhoods consistently generates 0.0% matching errors over all frame separation levels and outperforms all other methods we

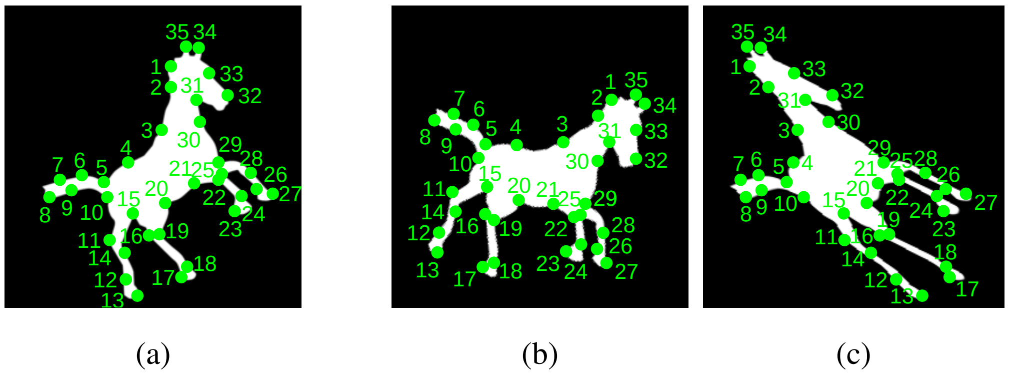


Fig. 7

EXAMPLE MATCHING CASES FROM THE HORSE "ROTATION" AND "SHEAR" SEQUENCES USING SHAPE CONTEXT [5] FEATURES. (A) THE 1ST FRAME IN BOTH SEQUENCES. (B) THE 91ST FRAME IN THE "ROTATION" SEQUENCE. LABELED LANDMARKS IN IT ARE USED AS SCENE POINTS. MATCHING RESULTS ARE SHOWN AS NUMBERS. (C) THE 91ST FRAME IN THE "SHEAR" SEQUENCE. LABELED LANDMARKS IN IT ARE USED AS SCENE POINTS. MATCHING RESULTS ARE SHOWN AS NUMBERS. 0.0% MATCHING ERRORS ARE ACHIEVED IN BOTH CASES.

compared with. An example matching case by our method is shown in Fig. 5. The second row of Table I shows the matching errors by different methods on the *hotel* sequence. Our method again outperforms all other methods we compared with and generates a 1.04% average matching error. As shown in the example in Fig. 6.(b), all matching errors by our method are caused by mismatching the 17th and the 23rd template points, which have very similar features and are spatially close to each other.

B. Horse Rotation and Shear Sequences [10]

We followed the experimental setup in [10]. A 35-point-set labeled on the silhouette of a horse [4] is obtained from [10]. The "rotation" sequence is generated by rotating the horse point set by 90 degrees, and the "shear" sequence is generated by shearing it horizontally to twice its width. Each sequence consists of 299 frames. We created matching cases similarly to the previous experiment as pairs of images separated by a specific number of frames. Although the Shape Context feature is not transformation invariant, it was still used in this experiment to create more challenging matching cases. We set $w = 1$ because this experiment also looks for exactly one-to-one matchings. Since these two sequences contain only global transformation, we

TABLE II

MATCHING ERRORS (THE PERCENTAGE OF WRONG MATCHINGS) OF MATCHING CASES IN SECTION III-C BY OUR PROPOSED METHOD.

Matching Case	Our Method+DT	Our Method + kNN ($k = 5$)	Our Method + kNN ($k = 9$)	Our Method + kNN ($k = 13$)
$h\% = 10\%$	0.64%	0.67%	0.62%	0.62%
$h\% = 20\%$	2.95%	2.88%	2.97%	2.97%
$h\% = 30\%$	10.8%	10.9%	10.9%	10.8%
$h\% = 40\%$	21.9%	22.0%	22.1%	23.0%
$h\% = 50\%$	42.0%	42.0%	41.7%	41.8%

set a larger weight to the geometric cost term, $\lambda = 1.0$. However, unlike the previous experiment, neighborhoods defined by Delaunay Triangulation resulted in better matching performance than kNN in this experiment. This is because Delaunay Triangulation is more likely to take template points on the convex hull as neighbors even though they are far away from each other, which better enforces a global transformation between template points and matched scene points. Matching errors by different methods are shown in the third and fourth rows of Table I. Our method with Delaunay Triangulation outperforms all other compared methods and achieves 0.0% matching errors on both sequences. Our method with kNN also achieves 0.0% matching errors on all frame separation levels except the 90 one for the "rotation" sequence. Example matching cases from both sequences are shown in Fig. 7.

C. Synthetic Data with Missing Points

To test our method's robustness when template points' corresponding points are either not detected or occluded in the scene image, we created an experiment with random points using Shape Context [5] as features. For each matching case, we uniformly spread random points in the region $[100, 500] \times [100, 500]$ as template feature points. To generate scene feature points, we deleted $h\% \times n_t$ number of points from the template point set to simulate the effect of feature point mis-detection or occlusion, and added $h\% \times n_t$ number of randomly spread points in $[0, 600] \times [0, 600]$ as outliers. For each 10%, 20%, 30%, 40% and 50% occlusion and outlier level, we created 100 matching cases and matched them using our proposed method. We measured matching errors as the percentage of wrong matchings of undeleted template points. The statistics

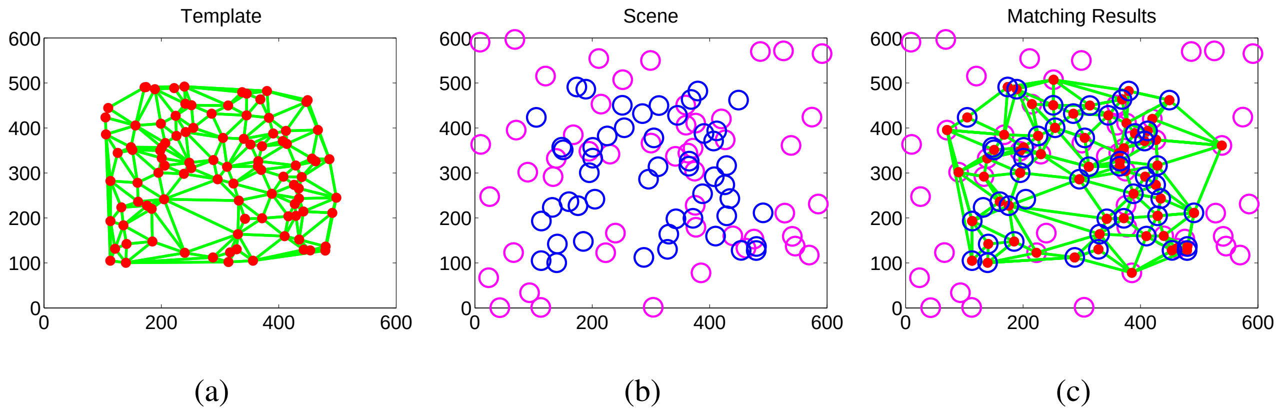


Fig. 8

AN EXAMPLE MATCHING CASE FROM SECTION III-C WITH NEIGHBORHOODS DEFINED BY DELAUNAY TRIANGULATION. (A) TEMPLATE POINTS (RED DOTS) WITH NEIGHBORHOODS DEFINED BY DELAUNAY TRIANGULATION (GREEN LINES). (B) SCENE POINTS WITH 50% UNDELETED POINTS (BLUE CIRCLES) AND 50% OUTLIER POINTS (MAGENTA CIRCLES). (C) MATCHING RESULTS BY OUR PROPOSED METHOD. 3 OUT OF 50 UNDELETED TEMPLATE POINTS ARE WRONGLY MATCHED.

of errors on the matching cases are shown in Table II. Our method is able to correctly match most undeleted template feature points even when $50\% \times n_t$ template points are deleted and $50\% \times n_t$ random points are added as outliers. One example matching case of 50% occlusion and outlier level is shown in Fig. 8. As illustrated in the matching results (Fig. 8.(c)), if a template points is not detected or occluded, our method tends to match it to another template point that best preserves the geometric properties of that deleted template point.

D. INRIA Datasets [30]

In our next experiment, we tested 4 sets (*boat*, *bark*, *graf*, and *wall*) of images from the INRIA datasets used in [30]. Each set contains 6 images. The *boat* and *bark* sets contain images undergoing scaling and rotation of natural scenes, while the *graf* and *wall* contain images of planar walls taken from different viewpoints. We created template points from the 1st frames of each set (Fig. 9) and matched them to the remaining images in the four sets. The ground truth of transformation parameters are provided with the images. To measure the matching errors, we first transformed template points with their ground-truth transformation parameters. The distance between each transformed template point and its matched scene point is calculated. If such a

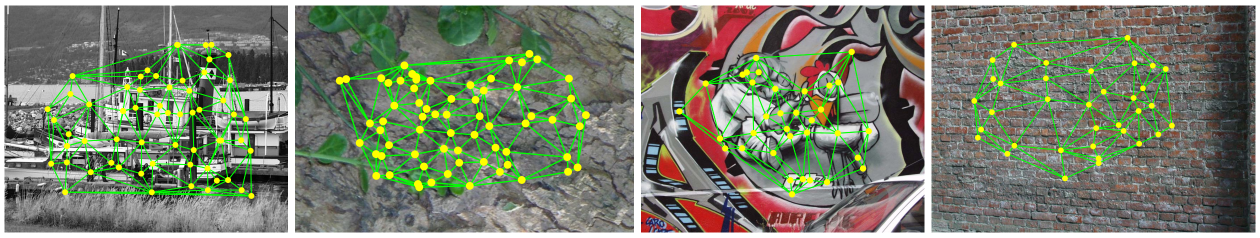


Fig. 9

THE 1ST FRAMES IN THE (A) *boat*, (B) *bark*, (C) *graf*, AND (D) *wall* SETS FROM THE INRIA DATASETS [30]. TEMPLATE POINTS ARE SHOWN AS YELLOW DOTS. NEIGHBORHOODS DEFINED BY DELAUNAY TRIANGULATION ARE SHOWN AS GREEN LINES.

distance is greater than 1.5 pixel, this matched scene point is counted as a wrong matching. The total number of mismatched points is then calculated as matching errors. Note that matching errors result from two aspects: (i) some template points' corresponding feature points are not detected in the scene image; (ii) wrong matchings caused by matching methods.

For the template points in the *boat* and *bark* sets, we chose them as SIFT feature points in the central area with scales greater than 6. This strategy is used to increase their corresponding points' probabilities of being detected in the scaled images. For the template points in the *graf* and *wall* sets, we used inerest points detected by MSER [28] in the central area of the first image. The two parameters of MSER, minimal region size and minimal margin, were set to 30 and 15, respectively. We further excluded duplicate points and points with scales less than 2. SIFT descriptor is used to calculate feature vectors at those detected salient locations. For the feature point in the scene images, we used SIFT descriptors' and MSER detectors' default parameter settings. We tested our method using the two different neighborhoods: Delaunay Triangulation and kNN with $k = 5$, 9 and 13. The LP method proposed in [21] is used for comparison. Because the code of [21] has a pre-set scaling range, it was not tested on the 4th-6th images in the *bark* and *boat* sets.

Matching results by our method with Delaunay Triangulation defined neighborhoods are shown in Fig. 10. The matching errors of tested methods are shown in Table III. The two different neighborhoods give similar matching performance on these sets when used in our method. For

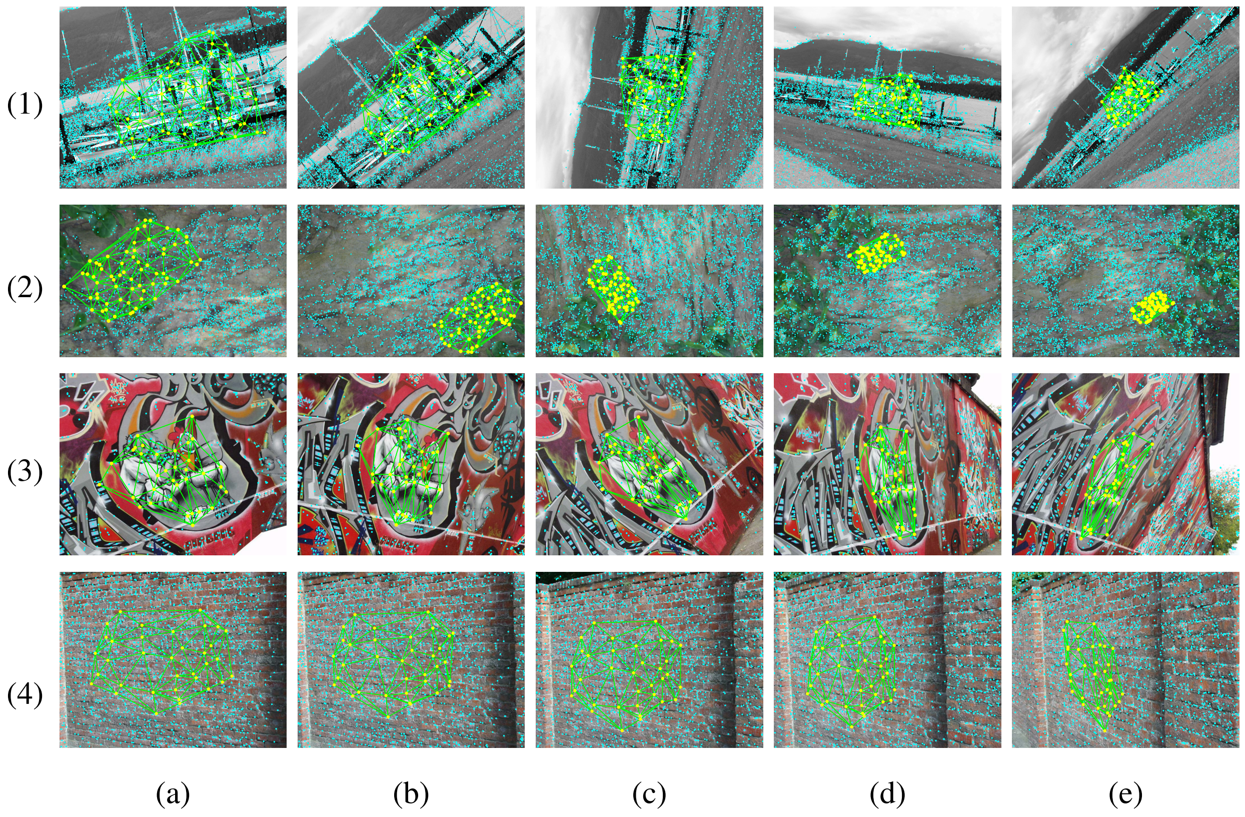


Fig. 10

MATCHING THE TEMPLATE POINTS IN FIG. 9 WITH NEIGHBORHOOD DEFINED BY DELAUNAY TRIANGULATION TO OTHER IMAGES IN THE (1) *boat*, (2) *bark*, (3) *graf*, AND (4) *wall*. UNMATCHED SCENE FEATURE POINTS ARE MARKED IN LIGHT BLUE. (A)-(E) THE 2RD TO THE 6TH FRAMES IN THE FOUR SETS.

the *boat* and *bark* sets, since the images are only undergoing similarity transformations, our method has similar performance as the LP method in [21]. The results of the two sets showed that our neighborhood structures are able to handle large scaling. For the *graf* and *wall* sets, our method outperforms the LP method in [21] because our method's locally affine constraints can better tolerate complex deformations.

E. Rotated and Occluded Objects in Cluttered Background

We modeled an *IEEE Spectrum* magazine (Fig. 1.(a)) and matched it to its transformed instances in scene images with cluttered background (Fig. 11). For the template point set, points were selected as SIFT points with scales between 2 and 10, and their neighborhoods are defined

TABLE III

MATCHING ERRORS (THE NUMBERS OF WRONG MATCHINGS) OF MATCHING CASES IN FIG. 10 BY OUR METHOD AND THE METHOD IN [21].

Matching Case	The Method in [21]	Our Method+DT	Our Method+kNN ($k = 5$)	Our Method+kNN ($k = 9$)	Our Method+kNN ($k = 13$)
boat 2 (Fig. 10.(1a))	29 out of 64	30 out of 64	30 out of 64	30 out of 64	30 out of 64
boat 3 (Fig. 10.(1b))	27 out of 64	30 out of 64	30 out of 64	30 out of 64	30 out of 64
boat 4 (Fig. 10.(1c))	n/a	40 out of 64	40 out of 64	40 out of 64	41 out of 64
boat 5 (Fig. 10.(1d))	n/a	42 out of 42	40 out of 64	42 out of 64	42 out of 64
boat 6 (Fig. 10.(1e))	n/a	55 out of 64	55 out of 64	55 out of 64	55 out of 64
bark 2 (Fig. 10.(2a))	38 out of 67	39 out of 67	39 out of 67	39 out of 67	39 out of 67
bark 3 (Fig. 10.(2b))	60 out of 67	60 out of 67	60 out of 67	60 out of 67	60 out of 67
bark 4 (Fig. 10.(2c))	n/a	19 out of 67	19 out of 67	19 out of 67	19 out of 67
bark 5 (Fig. 10.(2d))	n/a	13 out of 67	13 out of 67	13 out of 67	13 out of 67
bark 6 (Fig. 10.(2e))	n/a	45 out of 67	45 out of 67	45 out of 67	45 out of 67
graf 2 (Fig. 10.(3a))	10 out of 34	10 out of 34	10 out of 34	10 out of 34	10 out of 34
graf 3 (Fig. 10.(3b))	31 out of 34	7 out of 34	8 out of 34	7 out of 34	7 out of 34
graf 4 (Fig. 10.(3c))	31 out of 34	8 out of 34	8 out of 34	8 out of 34	10 out of 34
graf 5 (Fig. 10.(3d))	33 out of 34	12 out of 34	14 out of 34	12 out of 34	13 out of 34
graf 6 (Fig. 10.(3e))	34 out of 34	16 out of 34	16 out of 34	16 out of 34	16 out of 34
wall 2 (Fig. 10.(4a))	20 out of 35	10 out of 35	10 out of 35	10 out of 35	10 out of 35
wall 3 (Fig. 10.(4b))	27 out of 35	7 out of 35	8 out of 35	7 out of 35	7 out of 35
wall 4 (Fig. 10.(4c))	29 out of 35	8 out of 35	8 out of 35	8 out of 35	10 out of 35
wall 5 (Fig. 10.(4d))	31 out of 35	12 out of 35	14 out of 35	12 out of 35	13 out of 35
wall 6 (Fig. 10.(4e))	35 out of 35	16 out of 35	16 out of 35	16 out of 35	16 out of 35

by Delaunay Triangulation. Although there were many outlier feature points (> 1000) in the scene images, and some template points' corresponding scene points were not detected or intensionally occluded (Fig. 11.(d)), our method still was able to match the magazine to the scene images robustly.

F. Objects Undergoing Articulated Deformations

Our local geometric constraint only tries to maintain each point's local geometric properties and thus can match objects undergoing articulated deformations. In Fig. 12, we show an experiment of matching a toy worm with distinctive features (Fig. 12.(a)) to its bended instances in scene images. To define the neighborhoods for template points, we manually removed some edges after calculating the Delaunay Triangulation of template points to avoid building strong

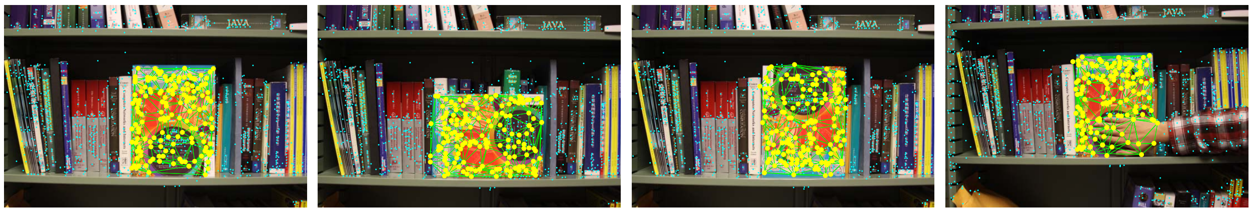


Fig. 11

MATCHING THE *Spectrum* MAGAZINE (FIG. 1) IN SCENE IMAGES WITH CLUTTERED BACKGROUND. (A) THE MAGAZINE IS NOT ROTATED. (B) THE MAGAZINE IS ROTATED BY 90 DEGREES. (C) THE MAGAZINE IS ROTATED BY 180 DEGREES. (D) THE MAGAZINE IS PARTIALLY OCCLUDED BY A HAND. UNMATCHED SCENE FEATURE POINTS ARE MARKED IN LIGHT BLUE.

connections between different moving parts. Results in Fig. 12.(c) and 12.(d) demonstrate the advantages of our local geometric constraint over the global constraint proposed in [21].

G. Real Videos

We did experiments on real videos, two taken by ourselves (the *Computer* and *Spectrum* magazine videos) and two obtained from the YouTube (the *butterfly* and *honeybee* videos). Similar to the matching experiments in Section III-E, we used SIFT points in the selected object regions as template points and built their neighboring connections through Delaunay triangulation (Fig. 13.(a)). We applied our method to every single frame of those videos and did not utilize any temporal information. The algorithm does not need initialization and can track an object undergoing large and complex deformations. We compared our method with the LP based method in [21] using those videos.

The *Computer* magazine and *butterfly* videos consist of mostly similarity transformations, with local deformations and some occlusions (Fig. 14.(1) and 14.(2)). For these two videos, our method has similar matching accuracy as the LP method in [21] (Fig. 13.(1) and 13.(2)) but has an asymptotically faster running speed.

The *Spectrum* magazine video consists of mostly affine transformations and non-rigid deformations (Fig. 14.(3)). On this video, our method outperformed the LP method in [21] because our geometric constraint is affine-invariant, and its local property enables it to handle larger non-rigid deformations. One such example is shown in Fig. 13.(3) where the magazine is wrapped

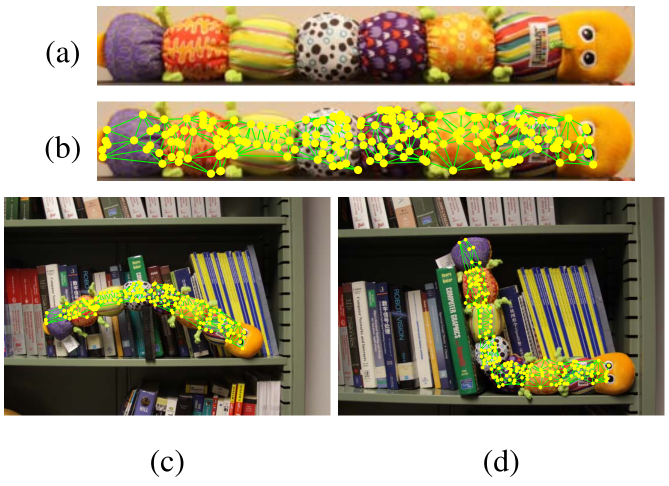


Fig. 12

MATCHING A TOY WORM UNDERGOING ARTICULATED DEFORMATIONS. (A) THE ORIGINAL IMAGE OF THE TOY WORM. (B) THE TEMPLATE POINTS AND THEIR NEIGHBORHOOD RELATIONSHIPS. (C) AND (D) TWO EXAMPLES OF MATCHING THE TOY WORM MODEL TO ITS INSTANCES THAT HAVE UNDERGONE ARTICULATED DEFORMATIONS.

inwards. The global geometric constraint of [21] prefers scaling the template point set globally. Our local constraint tries to maintain each point's local geometric properties so it can better handle such non-rigid deformations.

The *honeybee* video looks simple, but it has fewer distinctive feature points than the other videos which makes matching the honeybee a more challenging task (Fig. 14.(4)). Our method outperformed the LP method in [21] when a large portion of corresponding feature points are missing in the scene images. Fig. 13.(4) shows such an example where only a fraction of the feature points on the honeybee's tail part were detected. The global geometric constraint of the LP method in [21] favors all matched scene points maintaining a similar geometric structure as the template points. It matches part of the tail correctly but wrongly matches other parts to the background (Fig. 13.(4c)). In contrast, our geometric constraint only tries to keep local geometric structures and thus can match disappeared feature points to shrunken neighborhoods. The result by our method is shown in Fig. 13.(4d) where the tail part is correctly matched.

IV. LIMITATIONS AND CONCLUSIONS

Distinctive feature points. Although our method allows more complex geometric transfor-

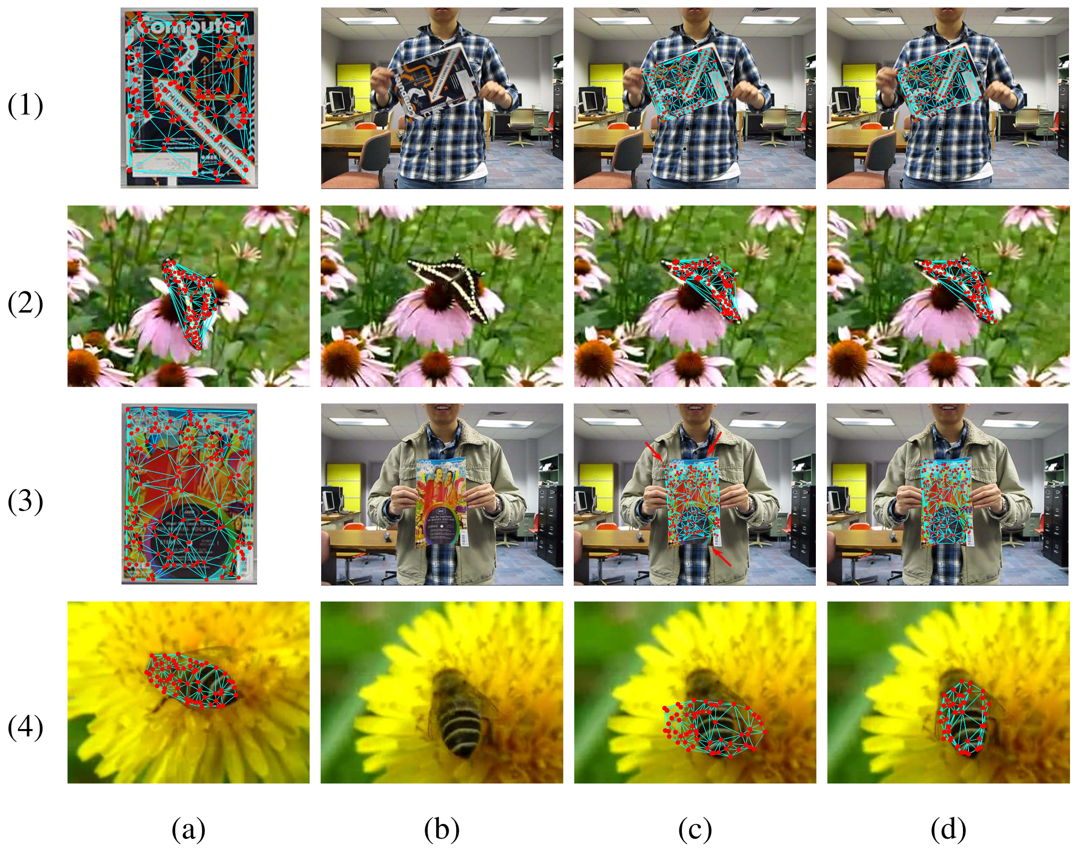


Fig. 13

SAMPLE COMPARISON RESULTS BY OUR METHOD AND THE LP METHOD IN [21] ON VIDEOS. (A) TEMPLATE POINTS. (B) SCENE FRAMES. (C) MATCHING RESULTS BY THE LP METHOD IN [21]. (D) MATCHING RESULTS BY THE PROPOSED METHOD. VIDEO RESULTS: HTTP://WWW.YOUTUBE.COM/WATCH?V=QZPYP0DTENA&LIST=PL5315098DD6D1F04C

mations, in our experiments we observed that our method also requires more distinctive feature vectors than the LP method in [21] does. If an object has fewer distinctive feature points and undergoes only similarity transformation, the LP method in [21] would outperform our method. One such example is the *bear* sequence in [21]. Two factors contribute to this phenomenon: (i) the optional constraint (12), which prevents the method from matching too many template points to one scene point, becomes less effective in the relaxed model in the continuous domain (11). Therefore, when features are not distinctive, template points may tend to match to only a few scene points to primarily minimize the geometric cost. (ii) Our locally affine invariant allows more freedom on geometric transformations, and provides weaker constraints when matching an

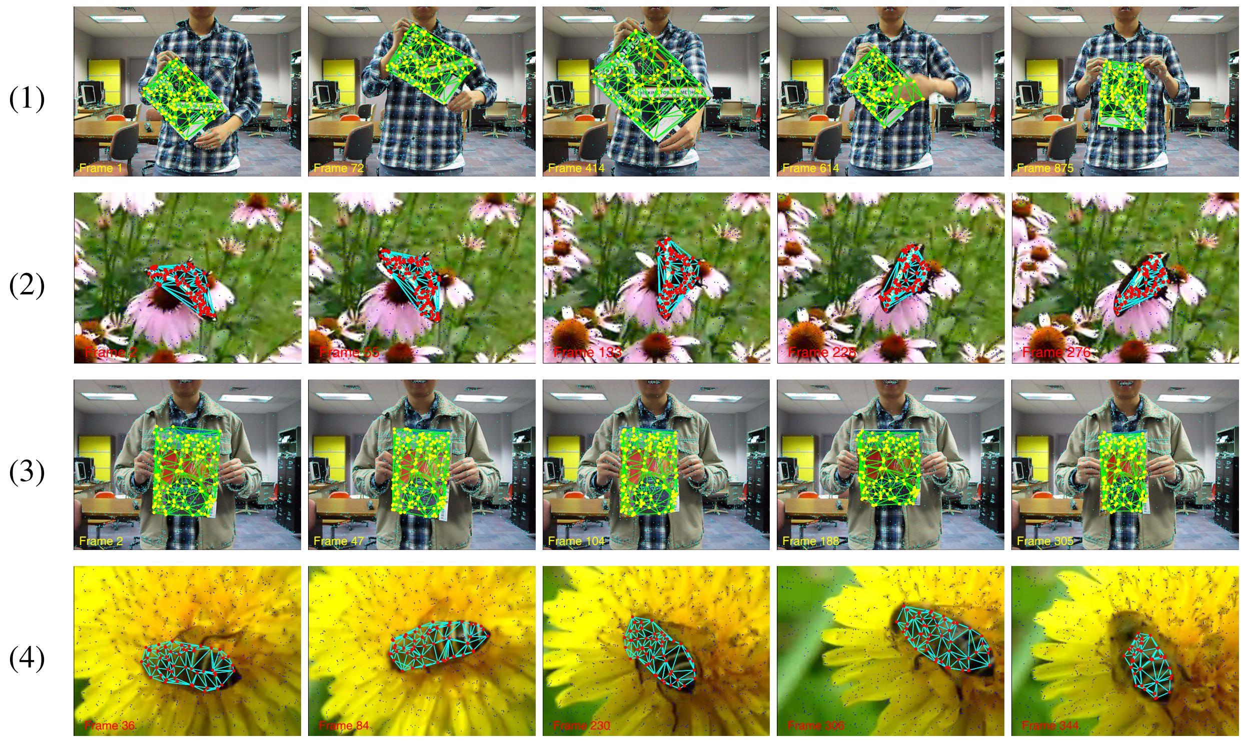


Fig. 14

SAMPLE MATCHING RESULTS BY OUR METHOD FROM (1) THE *Computer* MAGAZINE SEQUENCE, (2) THE *butterfly* SEQUENCE, (3) THE *Spectrum* MAGAZINE SEQUENCE, AND (4) THE *honeybee* SEQUENCE. UNMATCHED SCENE FEATURE POINTS ARE MARKED IN BLUE.

object undergoing only similarity transformation.

Occlusion handling remains a challenging problem for the graph matching [23], [16] and the LP based matching [20], [21] frameworks. Unlike the RANSAC methods, which can easily determine outliers as those violating a global transformation model, feature matching methods allow local deformations and therefore have difficulties determining occluded template points. Moreover, determining occluded points in the matching process may require additional binary variables, which makes this NP-hard problem even more difficult.

Appropriate weights in the objective function (11) are application-dependent and should be set case by case. However, some pre-processing steps, such as normalizing feature matching costs, can ease the search of appropriate weights. For a specific application, learning techniques [10], [24] can be used to determine the best weights.

In this paper, we presented a novel locally affine-invariant constraint for the LP-based object matching framework. This constraint depends on exactly representing each point by an affine combination of its neighboring points. Such representations were proved to be exact and can be easily solved by least squares. Our proposed constraint showed several advantages over those in previous works. Experiments on various matching cases for rigid and non-rigid objects demonstrated the effectiveness and efficiency of our proposed algorithm.

ACKNOWLEDGMENTS

We would like to thank Chenyang Xu (Siemens TTB), Hao Jiang (Boston College), and Paulo Mendonca (GE Global Research) for their valuable suggestions and comments. We are grateful to Tiberio Caetano and Julian McAuley (NICTA) for providing labeling data and raw results of their experiments in [10].

REFERENCES

[1] "lpsolve: sourceforge.net/projects/lpsolve."

[2] "CMU "hotel" data set, http://vasc.ri.cmu.edu/idb/html/motion/hotel/index.html," 2006.

[3] "CMU "house" data set, http://vasc.ri.cmu.edu/idb/html/motion/house/index.html," 2006.

[4] "Mythological Creatures 2D database, http://tosca.cs.technion.ac.il,," 2009.

[5] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[6] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 26–33, 2005.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[8] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE Trans. Pattern Analayis and Machine Intelligence*, vol. 26, no. 4, pp. 515–519, 2004.

[9] T. S. Caetano, T. Caelli, D. Schuurmans, and D. A. C. Barone, "Graphical models and point pattern matching," *IEEE Trans. Pattern Analayis and Machine Intelligence*, vol. 28, no. 10, pp. 1646–1662, 2006.

[10] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, pp. 2349–2374, 2009.

[11] M. Carcassoni and E. Hancock, "Spectral correspondence for point pattern matching," *Pattern Recognition*, vol. 36, pp. 193–204, 2003.

[12] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation," *IEEE Trans. Pattern Analayis and Machine Intelligence*, vol. 17, no. 8, pp. 749–764, 1995.

[13] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, pp. 114–141, 2003.

[14] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," *Proc. Neural Information Processing Systems*, pp. 313–320, 2006.

[15] A. D. J. Cross and E. R. Hancock, "Graph matching with a dual-step em algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1236–1253, 1998.

[16] O. Duchenne, F. Bach, I. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

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

[18] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, pp. 377–388, 1996.

[19] S. S. H. Jiang, T.-P. Tian, "Scale and rotation invariant matching using linearly augmented tree," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

[20] H. Jiang, M. S. Drew, and Z. Li, "Matching by linear programming and successive convexification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, pp. 959–975, 2007.

[21] H. Jiang and S. X. Yu, "Linear solution to scale and rotation invariant object matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

[22] J. V. Kittler and E. R. Hancock, "Combining evidence in probabilistic relaxation," *Intl Journal of Pattern Recognition and Artificial Intelligence*, vol. 3, pp. 29–51, 1989.

[23] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 1482–1489, 2005.

[24] ——, "Unsupervised learning for graph matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

[25] S. Z. Li, "A markov random field model for object matching under contextual constraints," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 866–869, 1994.

[26] H. Liu and S. Yan, "Common visual pattern discovery via spatially coherent correspondences," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

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

[28] J. Matas, O. Chum, M. Urba, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," *Proc. British Machine Vision Conference*, pp. 384–396, 2002.

[29] B. T. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Trans. Pattern Analayis and Machine Intelligence*, vol. 20, no. 5, pp. 493–503, 1998.

[30] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *Int'l J. Comp. Vis.*, vol. 60, pp. 63–86, 2004.

[31] M. Pelillo, "Replicator equations, maximal cliques, and graph isomorphism," vol. 11, pp. 1933–1955, 1999.

[32] R. Raguram, J. Frahm, and M. Pollefeys, "A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus," *Proc. European Conf. Computer Vision*, vol. 2, pp. 500–513, 2008.

[33] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. Academic Press.

[34] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[35] C. Schellewald, "Convex mathematical programs for relational matching of object views," *PhD Dissertation, Univ. of Mannhein*, 2004.

[36] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, pp. 530–535, 1997.

[37] L. Shapiro and J. Brady, “Feature-based correspondencełan eigenvector approach,” *Image and Vision Computing*, vol. 10, pp. 283–288, 1992.

[38] M. Torki and A. Elgammal, “One-shot multi-set non-rigid feature-spatial matching,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[39] L. Torresani, V. Kolmogorov, and C. Rother, “Feature correspondence via graph matching: Models and global optimization,” *Proc. European Conf. Computer Vision*, 2008.

[40] M. van Wyk, T. Durrani, and B. van Wyk, “A RKHS interpolator-based graph matching algorithm,” *IEEE Trans. Pattern Analayis and Machine Intelligence*, vol. 24, no. 7, pp. 988–995, 2002.

[41] H. Wang and E. R. Hancock, “A kernel view of spectral point pattern matching,” *Proc. Int’l Workshop Advances in Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition*, 2004.

[42] R. C. Wilson and E. R. Hancock, “Structural matching by discrete relaxation,” *IEEE Trans. Pattern Analayis and Machine Intelligence*, vol. 19, no. 6, pp. 634–648, 1997.

[43] R. Zass and A. Shashua, “Probabilistic graph and hypergraph matching,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[44] 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,” *Proc. Int’l Conf. Information Processing in Medical Imaging*, 2011.