Submission to GMP 2012

Features Matching Using Geometric Principles

Submission number: 87

Abstract

We present a feature matching algorithm for finding correspondences between two sets of features. Our approach is based on finding matching triangles and quadrangles, going beyond the single (point) and pairwise (segment) of features typically used. Our algorithm is formulated as supersymmetric-tensor-based higher-order matching scheme, the higher-order affinity measures taking into account matches between more than a single or pairwise features. We exploit a compact form of this affinity tensor, taking advantage of supersymmetry to devise an efficient sampling strategy to create the affinity tensor. Matching is performed using the rank-one approximation of the tensor directly with a higher-order power method. Experiments on both synthetic and real RGB images and depth videos show that our algorithm provides accurate and robust matching of feature correspondences in the complex scenario.

Keywords: Feature matching; Geometric principles; Supersymmetric tensor; High-order matching; Power method

1 Introduction

Finding correspondences between two sets of visual features (such as points of interest, edges, or regions) is a fundamental problem in many geometric computing, computer graphics and computer vision tasks, such as feature extraction [15, 3, 10, 18], shape matching [21, 2, 4, 17], and 3D registration [1] and reconstruction [9].

This challenging problem has attracted considerable attention. A feature itself may comprise one or more items of a given kind, so we may match single points to single points (point-single), point pairs separated by a fixed distance to other point pairs (segment-double), triples of points forming a triangle to other triples of points (triangle-triple), quadruples of points forming a quadrangle to other quadruples of points (quadrangle-quadruple), and so on. When single features are matched, we must solve a linear assignment problem, or if multiple features are matched at once, a quadratic or higher-order assignment problem results [6].

Linear assignment matches single features in one set with single features in the other set, and could be mainly treated as single points linkage. Matching two feature sets by considering similarities of *single* features from each set can easily fail in the presence of ambiguities such as repeated elements, textures or relatively uniform local appearance.

Quadratic and higher-order assignment matches groups of features in one set simultaneously with groups from the other set, and and requires a greater consistency between the information being matched, making it more reliable. As well as the features themselves, other constraints such as consistency of the distances between the features being matched are also enforced, greatly improving the reliability of higher order matching.

As a particular example of *quadratic* assignment, Leordanu and Hebert [14] consider pairs of feature descriptors, and distances between pairs of features from each set to reduce the number of incorrect correspondences. Such pairwise distance constraints are particularly helpful in cases when the features themselves have low discriminative ability.

Posing feature matching problems as *higher-order* assignment problems has attracted much recent interest. Higher-order assignment further generalizes the assignment problem to include yet more complex constraints between features. For example, a third-order potential function, proposed in [8, 5], quantifies the affinity between two point triples by measuring the similarity of the angles formed by such two feature tuples (triangles). However, this angular similarity value only calculates the total differences in corresponding angles, and does not change with the order of the input assignments. By changing the affinity tensor to a *supersymmetric* tensor [11], the limitation could be effectively solved by our algorithm.

Thus, by formulating the higher-order problem using a supersymmetric affinity tensor, we obtain an accurate and efficient higher-order matching algorithm, which works well when matching a moderate number of features using triples or quadruples of features. We make use of the supersymmetry of the affinity tensor, and determine correspondences by finding the rank-one approximation of the tensor using a more efficient power method.

The contributions of this paper include:

- We show how to define a compact higher-order supersymmetric affinity tensor to express geometric consistency constraints between tuples of features (for triangle and quadrangle).
- Based on the supersymmetry of the affinity tensor, we propose a higher-order power iteration method, which efficiently solves the matching problem.
- The affinity tensor is physically created by using a new efficient sampling strategy for feature tuples, which avoids sampling repetitive items, reduces the number of feature tuples to be sampled while ensuring the accuracy of the power iteration result.

Our experiments on both synthetic and real image data sets show that our method is accurate and robust, and has competitive computational cost compared to previous algorithms.

Section 4 describes the higher-order supersymmetirc affinity tensor and the power iteration method. Experiments are shown in Section 5 and conclusions drawn in Section 6.

2 Related work

Finding correspondences between two sets of discrete features, such as points, is a classical problem, and thus there is a large literature on the subject. Previous approaches can be classified into the which match single points to single points, those which match pairs of points to pairs of points, and so on.

Matching single points to single points, i.e. linear assignment problems, only consider affinity measures between two graph nodes, one from each set being matched, typically the feature distance between the two feature points. In concrete terms, the linear assignment problem

is posed as: find a mapping $f: P_1 \to P_2$, such that the optimal assignment $S^* = \arg\max \sum_{i \in P_1} A(i, j(i))$, where $A: N_1 \times N_2 \to R$ is the *affinity matrix*, and $A(i,j)$ measures the affinity between feature $i \in P_1$ and feature $j \in P_2$. Such affinity measures rely heavily on descriptors computed using local information around each feature point (e.g. shape contexts [2], SIFT [15], spin images [10], heat diffusion signatures [18]) in many computer vision and geometric computing tasks. It is apparent that point-to-point matching is weak in that wrong correspondences maybe readily be established.

Matching pairs of points in one set to pairs of points in the other set leads to a quadratic assignment problem. We now have an affinity matrix $A: N_1N_2 \times N_1N_2 \to R$, where $A(i,j)$ measures the compatibilities between two assignments $s_i = (f_i^1, f_{j(i)}^2)$ and $s_j = (f_i'^1, f_{j'(i')}^2)$, which takes into account both similarity of point features *and* Euclidean distance between the points in a pair. The quadratic assignment problem seeks to find a mapping $f: P_1 \to P_2$ which represents the optimal assignment. Unfortunately, this problem is NP-hard, but spectral relaxation techniques [14] can provide good approximate solutions.

Several higher-order constraints beyond pairwise potentials have been proposed. In general, we may define an m^{th}-order affinity measure T_m to capture the affinity associated with making m particular simultaneous assignments $s_{i_1} = (f_{i_1}^1, f_{j_1(i_1)}^2,)$, $\cdots$, $s_{i_m} = (f_{i_m}^1, f_{j_m(i_m)}^2)$. Such higher-order methods can significantly improve matching accuracy, but the higher-order assignment problem is again NP-hard, and various approximate methods have again been developed. Zass and Sashua [20] consider a probabilistic model of soft hypergraph matching. They reduce the higher-order problem to a first-order one by marginalizing the higher-order tensor to a one dimensional probability vector. Duchenne et al. [8] introduced a third-order tensor in place of an affinity matrix to represent affinities of feature triples, and higher-order power iteration was used to achieve the final matching. Chertok et al. [5] treat the tensor as a joint probability of assignments, marginalize the affinity tensor to a matrix, and find optimal soft assignments by eigendecomposition of the matrix. Wang et al. [19] also built a third-order affinity tensor and obtained a final matching by rank-one approximation of the tensor. Higher-order assignment problems typically require large amounts of memory and computational resources. By reducing the number of elements needed to represent the affinity measures, the above approaches can efficiently match large numbers (many hundreds or more) of features. However, these approaches sparsify the affinity information to some degree, leading to a reduction in matching accuracy. When matching two feature sets with large differences in scale, the matching results may become unstable.

A similar idea for 3D registration, called the 4-Points Congruent Sets method (4PCS), was proposed by Aiger et al. [1]. It is a fast and robust alignment scheme for 3D point sets that uses widely separated point tuples, providing resilience to noise and outliers. The key geometric idea is that 4PCS defines a ratio property which is preserved for planar 4-point sets under affine transformations. We use similar geometric rules in a more general way for feature matching.

3 Overview

Assume we are given two sets of feature points P_1 and P_2, with N_1 and N_2 points respectively. The correspondences between these two feature sets can be represented by an *assignment matrix* X of size $N_1 \times N_2$ whose elements are 0 or 1. $X(i,j) = 1$ when $i \in P_1$ matches $j \in P_2$ and $X(i,j) = 0$ otherwise. X can be row-wise vectorized to give an assignment vector $\boldsymbol{x} \in \{0,1\}^{N_1N_2}$.

Solving the higher-order matching problem is equivalent to find the optimal assignment vector $\boldsymbol{x}^* =< x_{i_1}, \cdots, x_{i_N} > \in \{0,1\}^{N_1N_2}$, satisfying

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} \sum_{i_1, \cdots, i_N} \mathcal{T}_N(i_1, \cdots, i_N) x_{i_1} \cdots x_{i_N}. \quad (1)$$

Here, i_n stands for an assignment (i_n', j_n'), $(i_n' \in P_1, j_n' \in P_2)$, and the product $x_{i_1} \cdots x_{i_N}$ is 1 only if all assignments $\{i_n\}_{n=1}^N$ are equal to 1, which means the tuple of features $(i_1', \cdots, i_N')$ from P_1 is matched correspondingly to the tuple of features $(j_1', \cdots, j_N')$ from P_2. $\mathcal{T}_N(i_1, \cdots, i_N)$ defines the affinity of the set of assignments $\{i_n\}_{n=1}^N$; it also can be seen as the affinity measure between the ordered feature tuples $(i_1', \cdots, i_N')$ and $(j_1', \cdots, j_N')$.

In the paper, we consider the one-to-many correspondence problem. We assume that a point in P_1 is matched to exactly one point in P_2, but that the reverse is not necessarily true. If *do* we want to treat both datasets in the same way, we can first match P_1 to P_2, then match P_2 to P_1, and then combine the matching results by taking their union or intersection. Uniqueness of matches for P_1 means that the assignment matrix X satisfies $\sum_i X(i,j) = 1$.

From Equ.(1) we can see that there are four issues to be considered when using higher-order matching algorithms. How should we:

- organize and express the affinity measures $\mathcal{T}_N$? (see Section 4.1)
- approximately solve the optimal higher-order assignment problem efficiently? (see Section 4.2)
- define the affinity measure between two feature tuples, or equivalently, the higher-order potential function ϕ_N? (see Section 4.3)
- determine an appropriate sampling strategy that physically build the affinity tensor and influence good matching accuracy? (see Section 4.4)

4 Feature matching

We now discuss the first and second issues mentioned above, which are independent of application; later we turn to definition of affinity measure, which is application dependent, and sampling strategy.

4.1 Supersymmetric affinity tensor

A tensor generalises vectors and matrices to higher dimensions: a vector is a tensor of order one, and a matrix is a tensor of order two. A higher-order tensor can be expressed as a multi-dimensional array [12]. Here we consider a higher-order supersymmetric affinity tensor, which represents a real-valued higher-order affinity between feature tuples.

Definition 1 (Supersymmetric Tensor)**.** *A tensor is called supersymmetric if its entries are invariant under any permutation of its indices [11].*

For example, a third-order supersymmetric tensor $\mathcal{T}_3$, satisfies the relationships: $\mathcal{T}_3(i_1,i_2,i_3) = \mathcal{T}_3(i_1,i_3,i_2) = \mathcal{T}_3(i_2,i_1,i_3) = \mathcal{T}_3(i_2,i_3,i_1) = \mathcal{T}_3(i_3,i_1,i_2) = \mathcal{T}_3(i_3,i_2,i_1)$.

Consider $T_N(i_1, \cdots, i_N)$, in Equ.(1), which measures the affinity of the assignments $(i_1, \cdots, i_N)$; in other words it uses the value $\phi_N(i_1, \cdots, i_N)$ to give a score when matching the ordered feature tuple $(i_1', \cdots, i_N')$ from P_1

to the ordered feature tuple $(j_1^{'}, \cdots, j_N^{'})$ from P_2,. Often, the value of $\phi_N(i_1, \cdots, i_N)$ is invariant under permutation of its arguments $(i_1, \cdots, i_N)$. Obviously, a higher-order supersymmetric tensor $\mathcal{T}$ can be used to capture this information:

Definition 2 (Supersymmetric Affinity Tensor)**.** *Given two feature sets P_1 and P_2, with N_1 and N_2 features respectively, the supersymmetric affinity tensor is an N^{th} order $I_1 \cdots \times I_N$, nonnegative tensor $\mathcal{T}_N$, where $I_1 = I_2 = \cdots = I_N = \{1, \cdots, N_1 N_2\}$, for which there exists a set of indices θ_N, and an N^{th} order potential function ϕ_N, such that*

$$\mathcal{T}_N(i_1, \ldots, i_N) = \begin{cases} \phi_N(\Omega(i_1, \ldots, i_N)) & ,\forall (i_1, \ldots, i_N) \in \theta_N \\ 0 & ,\forall (i_1, \ldots, i_N) \notin \theta_N \end{cases} \tag{2}$$

where Ω stands for an arbitrary permutation of the vector, and θ_N satisfies $\forall (i_1, i_2, \ldots, i_N) \in \theta_N, \forall i_p \in \{i_1, \ldots, i_N\}$ and $\forall i_q \in \{i_1, \ldots, i_N\} - \{i_p\}$ meets the requirement that $i_p \neq i_q$.

A tensor element with $(i_1, i_2, \ldots, i_N) \in \theta_N$ is called a potential element*, while other elements are called* non-potential element.

Using Definition 2, we now can greatly reduce the amount of storage needed, representing every potential element $\mathcal{T}_N(i_1, i_2, \ldots, i_N)$ by the canonical entry $\mathcal{T}_N(\text{sort}(i_1, i_2, \ldots, i_N)), \forall (i_1, i_2, \ldots, i_N) \in \theta_N$. Each stored value thus provides the value for $N!$ entries. As non-potential elements all have value zero, there is no need to store them. This greatly reduces the sampling needed for feature tuples when creating the affinity tensor, as discussed in Section 4.4. At the same time, it can be used to make the power iteration process more efficient: see Section 4.2.

4.2 Supersymmetric Tensor Based Higher-order Power Iteration Algorithm

Using Definition 2, Equ.(1) can be expressed as:

$$\begin{aligned} \boldsymbol{x}^* &= \arg\max_{\boldsymbol{x}} \sum_{i_1, i_2, \cdots, i_N} \mathcal{T}_N(i_1, \cdots, i_N) x_{i_1}, \cdots, x_{i_N} \\ &= \max < \mathcal{T}_N, \boldsymbol{x}^{\star N} > \end{aligned} \tag{3}$$

where $\star$ is called the Tucker product [11], and $\boldsymbol{x} \in \{0, 1\}^{N_1 N_2}$.

As noted, solving Equ.(3) is an NP-complete problem, so it is common to relax the constraints: the binary assignment vector $\boldsymbol{x} \in \{0, 1\}^{N_1 N_2}$ is replaced by an assignment vector $\boldsymbol{u}$ with elements taking real values in $[0, 1]$. This changes the optimization problem to one of computing the rank-one approximation of the affinity tensor $\mathcal{T}_N$ [13], i.e. finding a scalar λ and a unit norm vector $\boldsymbol{u} \in \mathbb{R}^{N_1 N_2}$, such that the tensor $\hat{\mathcal{T}_N} = \lambda \boldsymbol{u} \star \boldsymbol{u} \star \cdots \star \boldsymbol{u} = \boldsymbol{u}^{\star N}$ minimizes the function $f(\hat{\mathcal{T}_N}) = \|\mathcal{T}_N - \hat{\mathcal{T}_N}\|$. The final matching result is found by replacing each element of $\boldsymbol{u}$ by 0 or 1 according to whichever it is closer to.

The higher-order power method is commonly used to find the rank-one tensor approximation; a version for supersymmetric tensors (S-HOPM) is given in [11]. The S-HOPM algorithm converges under the assumption of convexity for the functional induced by the tensor [11], which is satisfied in many practical applications.

In their recent work, Duchenne et al. [8] failed to note that the whole iteration process can be simplified by taking advantage of supersymmetry. In [8], all tensor elements take part in the iteration process, which is unencessary. For example, given a third-order supersymmetric affinity tensor $\mathcal{T}_3$, an element $\phi_3(i, j, k)$ with index (i, j, k) and the element $\phi_3(i, k, j)$ with index (i, k, j) are the same, and so $\phi_3(i, k, j)$ can be reduced like $\phi_3(i, j, k)$. Redundant tensor elements also lead to a further problem. There is no guarantee in the method [8] that all tensor elements will be provided or all tensor elements will be balanced. For example, both elements $\phi_3(i, j, k)$ and $\phi_3(i, k, j)$ may occur, while just one element $\phi_3(i, j, l)$, $l \neq k$ may be present amongst the elements of $\mathcal{T}_3$. Such unbalanced redundant tensor elements disrupt the power iteration process, leading to incorrect results.

Algorithm 1 Higher-order power iteration method for a supersymmetric affinity tensor (with $\mathcal{C}_1$ norm)

Input: Nth-order supersymmetric affinity tensor $\mathcal{T}_n$
Output: unit-norm vector $\boldsymbol{u}$

1: Initialize $\boldsymbol{u}_0$, $k = 1$
2: **repeat**
3: **for** all $(i_1, i_2, \cdots, i_N) \in \theta_N$ **do**
4: **for** all $m \in (i_1, i_2, \cdots, i_N)$ **do**
5: $v_m^{(k)} = (N-1)! \cdot \phi_N(i_1, i_2, \cdots, i_N) \cdot 2 v_m^{(k-1)} v_{i_1}^{2(k-1)} \cdots v_{m-1}^{2(k-1)} v_{m+1}^{2(k-1)} \cdots v_{i_N}^{2(k-1)}$
6: **end**
7: **for** $i = 1 : N_1$ **do**
8: $v^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2) = \hat{v}^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2) / \|\hat{v}^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2)\|_2$
9: **end**
10: **end**
11: $k = k + 1$;
12: **until** convergence;

Note: $v^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2)$ denotes the slice of $v^{(k)}$ with indices from $(i-1) \cdot N_2 + 1$ to $i \cdot N_2$.

We solve the above problems by proposing a new efficient higher-order power iteration algorithm, Algortihm 1, based on the supersymmetric affinity tensor. We rely on two ideas. First, we take advantage of the supersymmetry. Secondly, many of the elements of the affinity tensor are zero non-potential elements: it is much more efficient to perform the power iteration by just considering the non-zero potential elements.

For an Nth-order supersymmetric affinity tensor, the corresponding result is:

$$\begin{aligned} &\forall m \in (i_1, i_2, \cdots, i_N), \\ &v_m^{(k)} = (N-1)! \cdot \phi_N(i_1, i_2, \cdots, i_N) \cdot 2 v_m^{(k-1)} v_{i_1}^{2(k-1)} \cdots v_{m-1}^{2(k-1)} v_{m+1}^{2(k-1)} \cdots v_{i_N}^{2(k-1)} \end{aligned} \tag{4}$$

This version excludes each non-potential element from the iteration process, so is more efficient, and the complexity of the whole iteration process only depends on the number $|\theta_N|$ of affinities. Step 5 in Algorithm 1 includes all permutations of each potential element $\mathcal{T}_n(i_1, i_2, \cdots, i_n)$, which are expressed by a single potential function $\phi_n(i_1, i_2, \cdots, i_n)$. This method reduces memory costs while keeping accuracy: Algorithm 1 depends on all potential elements.

Many initialization schemes have been proposed for the power method [11]. We simply use positive random values to initialize u_0, which ensures that the algorithm converges.

4.3 Higher-order Potentials

Different higher-order potentials are appropriate for different applications. Here we give two higher-order potentials, one possessing invariance under similarity transformations (based on triangles) and the other possessing affine invariance (based on quadrilaterals), these kinds of invariance

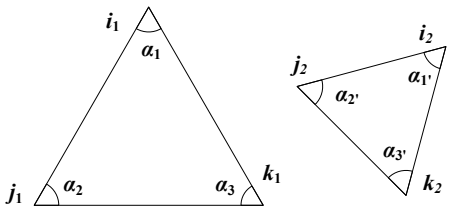


Figure 1: Third-order potential (triangle).

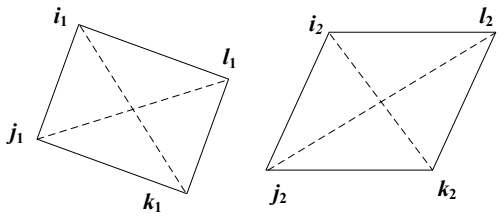


Figure 2: Fourth-order potential (quadrangle).

being useful for many correspondence problems in geometric computing. The potentials are based on a Gaussian kernel which guarantees the tensor elements are non-negative and invariant to any permutation of the input assignments.

First we restate a well-known [8, 5] third-order geometric-similarity invariant potential ϕ_3 linking two feature tuples, each having three features. Similarity of triangles formed by three points corresponds to invariance under scaling, rotation and translation—interior angles do not change. Thus ϕ_3 can be defined in terms of differences of corresponding interior angles:

$$\begin{aligned}\phi_3(i,j,k) &= \phi_3(\{i_1,i_2\},\{j_1,j_2\},\{k_1,k_2\}) \\ &= \exp(-1/\varepsilon^2 \sum\nolimits_{(l,l')} \|\alpha_l - \alpha_{l'}\|^2) \quad (5)\end{aligned}$$

where $\varepsilon > 0$ is the is the kernel bandwidth, and $\{\alpha_l\}_{l=1}^{3}$ and $\{\alpha_l^{'}\}_{l'=1'}^{3}$ are the angles triangles formed by feature triples (i_1, j_1, k_1) and (i_2, j_2, k_2): see Fig.1. Each point corresponds to one interior angle.

Feature image matching under more general affine transformations is also a common problem, so here we introduce a new fourth-order potential ϕ_4 which is affine-invariant, linking feature tuples with four features each. We use affine invariance of the ratio between two closed areas to define ϕ_4 as:

$$\begin{aligned}\phi_4(i,j,k,l) &= \phi_4(\{i_1,i_2\},\{j_1,j_2\},\{k_1,k_2\},\{l_1,l_2\}) \\ &= \exp(-1/\varepsilon^2 \sum\nolimits_{(l,l')} \|\delta_l - \delta_{l'}\|^2) \quad (6)\end{aligned}$$

where $\{\delta_l\}_{l=1}^{4}$ and $\{\delta_l^{'}\}_{l'=1'}^{4}$ are the ratios between the area of one triangle formed by three feature points and the area of the quadrilateral formed by all four feature points, so $\delta_1 = S_{\triangle i_1 j_1 k_1}/S_{\square i_1 j_1 k_1 l_1}$, $\delta_2 = S_{\triangle j_1 k_1 l_1}/S_{\square i_1 j_1 k_1 l_1}$, $\delta_3 = S_{\triangle i_1 k_1 l_1}/S_{\square i_1 j_1 k_1 l_1}$, $\delta_4 = S_{\triangle i_1 j_1 l_1}/S_{\square i_1 j_1 k_1 l_1}$, and similarly for the other quadrilateral.

We will use these two higher-order potentials to evaluate our algorithm.

4.4 Sampling strategy

Given two feature sets P_1 and P_2 with N_1 and N_2 features respectively, a potential element may be obtained by using two feature tuples sampled from each feature set separately. For Nth-order matching, a naive way to construct the potential elements is as follows: first find all feature tuples for P_1 and P_2, as F_1 and F_2; then $\forall(f_{i_1}^1, f_{i_2}^1, \cdots, f_{i_N}^1) \in F_1$, calculating the potentials for $(f_{i_1}^1, f_{i_2}^1, \cdots, f_{i_N}^1)$ with all feature tuples in F_2. This naive method is very expensive, which is why sampling is used. Choice of feature tuples to calculate the potentials directly determines the size $|\theta_N\|$ and influences the matching accuracy. Different sampling strategies can be chosen for different applications, using random sampling for general feature correspondence problems where there is no guidance to provide a better sampling method.

Here we use a random sampling approach. In order to cover all features in P_1 in F_1, we repeatedly take one feature as a required element, and then randomly choose t_1 feature tuples containing thise required element. We repeat this process until all features in P_1 have been chosen once as a required element. We do the same for P_2. We now have two feature-tuple sets for F_1 and F_2, containing $N_1 t_1$ and $N_2 t_2$ feature tuples separately.

Now, $\forall(f_{i_1}^1, f_{i_2}^1, \cdots, f_{i_N}^1) \in F_1$, we find the k most similar features in F_2 to build k potential elements as ϕ_i^k. Combining all the potential elements obtained, we form the desired potential element set $\theta_N = \{\phi_i^k\}_{i=1}^{N_1 t_1}$, with the size $|\theta_N| = N_1 t_1 k$. The parameters t_1, t_2 and k must be chosen according to the size of the feature sets. In practice for two point sets each with a hundred points, this approach works well when $t_1 \approx t_2 \approx 50$ and $k \approx 300$ for third-order or fourth-order matching. The sampling cost is $O(n\,t\,k \log n)$, where $n = \max(N_1, N_2), t = \max(t_1, t_2)$.

The most important part of the process is to use the supersymmetry of the affinity tensor. An N^{th}-order supersymmetric affinity tensor must satisfy:

$$\begin{aligned}&\forall(i_1, i_2, \cdots, i_N), (j_1, j_2, \ldots, j_N) \in \theta_N, \\ &\quad (i_1, i_2, \cdots, i_N) \neq \Omega(j_1, j_2, \cdots, j_N) \quad (7)\end{aligned}$$

where Ω is an arbitrary permutation. Thus, we use a sampling constraint that the sets of feature tuples F_1 and F_2 obtained from the sampling process, should have no repetition, in the sense that

$$\begin{aligned}&\forall(f_{i_1}^1, f_{i_2}^1, \cdots, f_{i_N}^1), (f_{j_1}^1, f_{j_2}^1, \cdots, f_{j_N}^1) \in F_1, \\ &\quad (f_{i_1}^1, f_{i_2}^1, \cdots, f_{i_N}^1) \neq \Omega(f_{j_1}^1, f_{j_2}^1, \cdots, f_{j_N}^1) \quad (8)\end{aligned}$$

$$\begin{aligned}&\forall(f_{i_1}^2, f_{i_2}^2, \cdots, f_{i_N}^2), (f_{j_1}^2, f_{j_2}^2, \cdots, f_{j_N}^2) \in F_2, \\ &\quad (f_{i_1}^2, f_{i_2}^2, \cdots, f_{i_N}^2) \neq \Omega(f_{j_1}^2, f_{j_2}^2, \cdots, f_{j_N}^2) \quad (9)\end{aligned}$$

and Ω is arbitrary permutation. This sampling constraint eliminates overlaps that may appear in the potential elements θ_N.

When N_1 is not equal to N_2 (suppose $N_1 < N_2$), we perform sampling as in [8], using random sampling for P_1 but full sampling for P_2 (to find all N_2^N feature tuples). However we still use the sampling constraint for feature-tuple set F_1, so our strategy is different to that in [8]. The process of building the tensor elements is the same to the process stated above when N_1 equals to N_2. In practice this works well when matching two feature sets with different numbers of features.

Earlier work [8, 20] also adopted random sampling, but failed to impose any constraint on the sampling process, leading to the possibility that feature tuples may include repetition. For example, for third-order matching, it is possible that a feature tuple $(f_{i_1}^1, f_{i_2}^1, f_{i_3}^1)$ may be sampled from P_1 and $(f_{i_1}^2, f_{i_2}^2, f_{i_3}^2)$ from P_2, and also a feature tuple $(f_{i_1}^1, f_{i_3}^1, f_{i_2}^1)$ sampled from P_1 and $(f_{i_1}^2, f_{i_3}^2, f_{i_2}^2)$ from P_2. That will create two tensor elements $\phi_3(s_{i_1}, s_{i_2}, s_{i_3})$ with index $(s_{i_1}, s_{i_2}, s_{i_3})$ and $\phi_3(s_{i_1}, s_{i_3}, s_{i_2})$ with index $(s_{i_1}, s_{i_3}, s_{i_2})$, which are the same. However, we just need one tensor element to express the affinity measure on the assignment group $(s_{i_1}, s_{i_2}, s_{i_3})$. This problem not only

makes the potential elements redundant but also affects the accuracy of the power iteration, because the numbers of each tensor element may not be equal and some elements may be used more than once during the iteration progress. Our method reduces the sampling cost, while also improving the accuracy of the power iteration.

5 Experiments

We have used synthetically generated random data as well as real images to evaluate our proposed method, and to provide comparisons to other state-of-art algorithms. The first experiment uses artificial data, while the other three use real image data for three classical correspondence problems: image matching under affine transformation, image matching on deformable surfaces, and image matching under projective transformations.

We used third-order feature matching for the artificial data experiment, for deformable surface matching and for projective image matching. Our fourth-order feature matching approach was applied to affine image matching.

We chose as a basis for comparison bipartite graph matching [2] (a first-order method), the spectral method [7] (a pairwise method), a third-order tensor method [8] and the hyper graph matching method [20], using the authors' code in each case. To enable direct comparison, the third-order tensor method [8], the hyper graph matching method [20] and our method used the same potential.

5.1 Artificial data

First we used artificial data to verify the performance of our method quantitatively, using a rotation test, a rescaling test, a distortion test and an outlier test.

For the first three tests, we generated 50 random points in the 2D plane for P_1, then P_2 was obtained by the following formula:

Rotation test:

$$P_2 = R_\alpha \cdot S_\delta \cdot P_1 + N(0, 0.05),\ \alpha \in \{0, 7\pi/4\}\ \text{ step } \pi/4, \forall S_\delta \in (0.5, 1.5)$$

Scaling test:

$$P_2 = R_{\delta'} \cdot S \cdot P_1 + N(0, 0.05),\ S \in \{1, 2, 4, 6, 8\}, \forall \delta' \in (-10^\circ, 10^\circ)$$

Distortion test:

$$P_2 = R_{\delta'} \cdot S_\delta \cdot P_1 + N(0, \sigma),\ \sigma \in [0, 1],\ \forall \delta' \in (-10^\circ, 10^\circ), \forall S_\delta \in (0.5, 1.5)$$

where S_δ and $R_{\delta'}$ are small random disturbances of scale and rotation angle in the ranges $(0.5, 1.5)$ and $(-10^\circ, 10^\circ)$ respectively. N, S and σ stand for Gaussian noise, the scaling factor and noise variance separately. In the rotation test, all points in P_1 were rotated through the same angle α around the origin. In the scaling test, all points in P_2 were scaled by the same factor S. P_2 in the distortion test was generated by perturbing the positions of all points in P_1 using Gaussian noise N.

For the outlier test, we generated 20 random points as P_1, and the point set P_2 was obtained by adding random numbers of outliers (from $0-60$ step 10, $60-100$ step 5) to P_1, then small random changes of scale and rotation were added, as well as Gaussian noise ($\sigma = 0.05$).

We compared our third order method with all four other methods. Each test was executed 50 times, and we measured the matching accuracy as the number of correctly matched points divided by the total number of points that could potentially be matched. The mean accuracy over all experiments is given in Figures 3(a)–3(d). We can see that no matter how large the changes made to P_1 (in terms of rotation, rescaling, distortion and outliers), our method could still find the correct correspondences, and outperforms other methods. For larger amounts of rotation and scaling, the bipartite graph matching method [2] soon fails, while higher-order methods perform much better due to the addition of geometric constraints. However, the third order method in [8] and the hyper graph matching method do not perform so well, because the geometric relationships among elements are not established accurately. It is also clear that the spectral method [7] cannot deal with large scale factors.

To provide a fair comparison, the number of tensor elements generated in our method, [8] and [20] were kept the same in all four tests (but different numbers were used for each method). In the first three tests, our method used 5000 feature tuples from P_1 and P_2 is 5000, while 150000 were used for [8] and 10000 for [20]. The number of feature tuples used in the outlier test were equivalent for our method, [8] and [20].

Matching two feature sets each with 50 features took an average of about 1.8s[1] for our method, 2.6s for [8], 0.9s for [20], 0.37s for [7], and 0.18s for [2]. As we use the same tensor size but sample fewer feature tuples, we achiev a higher matching accuracy with less computation than [8] and [20].

5.2 Image matching under affine transformation

Matching images related by an affine transformation is an important task. We used two publicly available image sets, `graf` and `wall`[2], to evaluate the matching accuracy of our method under affine transformations. These image sets each contain 6 images of a planar wall, which we numbered 1–6. For each image set, we used features generated from image 1 to match features in images 2 to 6 separately.

For the test image sets, we used 30 feature points detected by MSER [16] in the central area of image 1 to be feature set P_1 (the green points in Fig.4(b)). We then used the the transformation matrices provided with the image sets to find corresponding points for the feature set P_2 for images 2–6. In order to assess the robustness of the algorithms, we manually added outliers to P_1, shown as yellow points in Fig.4(b). We successively increased the number of outliers until the ratio of outliers was 1/3.

For this test, we compared our method with the spectral method [7] (a pairwise method), a tensor based method [8], and the hypergraph matching method [20], using the same fourth-order potentials for the higher-order methods. Method [8], [20] and ours used an equivalent tensor size. In each test, 6000 feature tuples were used in our method, 813000 for [8] and 12000 for [20].

In both image sets, the images were taken from quite different viewpoints, leading to quite different texture appearances. It is thus difficult to match features just using an MSER detector or a SIFT detector. The results in Fig.5 show that the pairwise method SMAC [7] as a result fails to provide a high matching accuracy, while the higher-order methods are much better as they take structural information into account. They work well if there are not too many outliers.

It is clear that our method is much more stable than the other higher-order methods. For the `graf` image set our method correctly matched the feature set P_1 to P_2 in images 2–6 up to the maximum outlier percentage. For image 6 of the `wall` set, our method still found most correct correspondences with 33.3% outliers. Although the tensor based method and the hyper graph matching method [20] performed well in the absence of outliers, their match-

[1] All methods were implemented in Matlab on a 2.3GHz Core2Duo with 2GB memory.

[2] From http://www.robots.ox.ac.uk/~vgg/data/data-aff.html

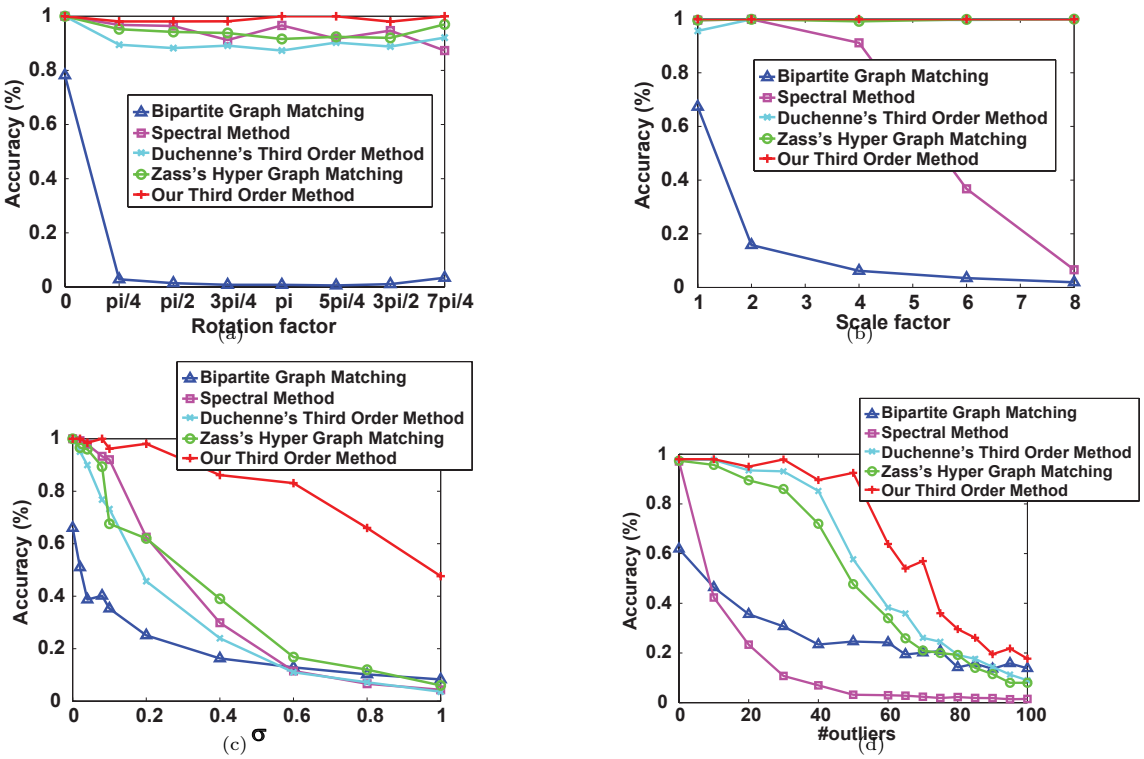


Figure 3: Fig.3(a) to 3(d), are the results of rotation test, rescaling test, distortion test and outlier test in the artificial data experiment.

Figure 4: Left: image 1 in `graf` set, right: image 1 in `wall` set. Model features detected by MSER in image 1 are shown in green; yellow points are outliers.

ing accuracy dropped rapidly as the outlier percentage increased.

5.3 Deformable surface matching

Registration of an object subject to large non-affine deformations is also an important problem. To assess the performance of our method, we established correspondences of feature points on the surface of a deforming object using our method, the spectral method [7], the tensor based method [8] and the hyper graph matching method [20]. Again, all higher-order methods used the same third-order potentials from Section **??** and all maintained an equivalent tensor size.

We chose four deformable surface image sets[3] containing a cloth, a curving piece of paper, a cushion and a creasing piece of paper. The surfaces of the cloth and the curving piece of paper underwent relatively smooth deformations, while the surfaces of the cushion and the creasing piece of paper included sharp folds.

[3]From http://cvlab.epfl.ch/data/dsr/

From each image set we randomly chose six frames before and after a large deformation. We randomly chose 100 corresponding points on each surface to be the features, using the provided ground truth. In each image set we chose the features on the frame most unlike the others (shown in Fig.7) as P_1 and matched them with the features in the other five frames.

The matching accuracy is given as the number of correctly matched points (according to the provided ground truth) divided by the total number of points that could potentially be matched. The results for all methods, using the four image sets, are given in Tables 1 and 2 and are illustrated in Fig.8. Our method matched all the features on the deforming surfaces, while others did not. In this test our method used 20000 feature tuples, while the method of [8] used 1010000 features and the method of [20] used40000. The average running time to match two feature sets each with 100 features was around 8s for our method, 13s for [8], 6.5s for [20], and 5s for [7]. As above it can be seen that our method is both efficient and much more

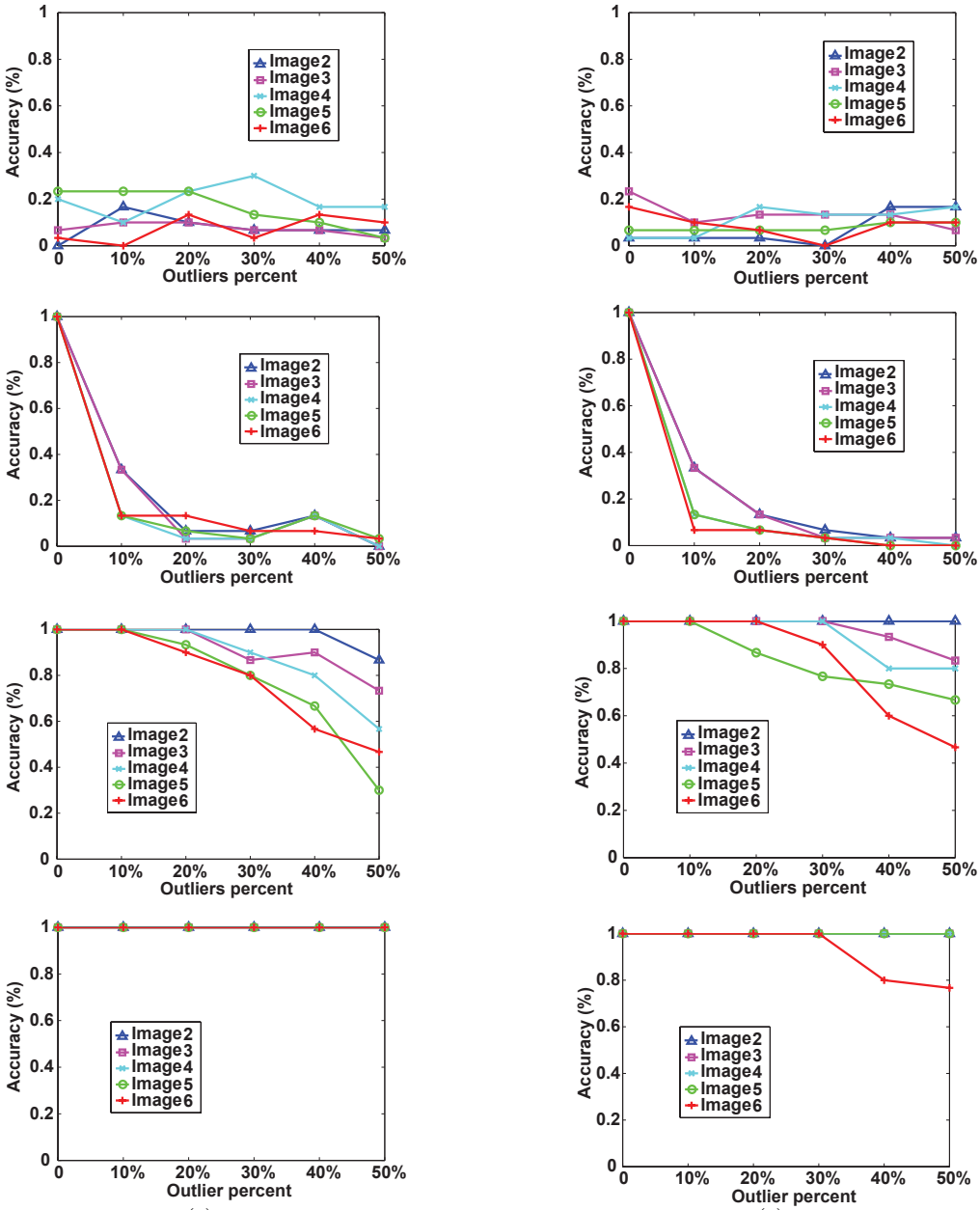


Figure 5: Accuracy as a function of percentage of outliers. Left: `graf` set, right: `wall` set. Top to bottom: results for the spectral method [7], the hyper graph matching method [20], the tensor based method [8] and our method.

Table 1: Error rate of deformable surface matching.

Dataset	*cloth*					*bending paper*				
Matching frames	*F88-F299*	*F107-F299*	*F120-F299*	*F258-F299*	*F305-F299*	*F262-F314*	*F275-F314*	*F280-F314*	*F289-F314*	*F301-F314*
Our method	0	0	0	0	0	0	0	0	0	0
[20]	2%	2%	2%	2%	2%	23%	9%	23%	18%	14%
[8]	25%	41%	30%	30%	42%	71%	67%	59%	55%	50%
[7]	93%	83%	77%	94%	90%	93%	98%	99%	93%	96%

Table 2: Error rate of deformable surface matching.

Dataset	*cushion*					*creased paper*				
Matching frames	*F144-F213*	*F156-F213*	*F165-F213*	*F172-F213*	*F188-F213*	*F309-F375*	*F320-F375*	*F330-F375*	*F340-F375*	*F352-F375*
Our method	0	0	0	0	0	0	0	0	0	0
[20]	8%	10%	2%	6%	5%	22%	10%	11%	2%	6%
[8]	61%	65%	59%	49%	30%	80%	74%	77%	68%	46%
[7]	95%	92%	93%	93%	92%	98%	94%	96%	92%	88%

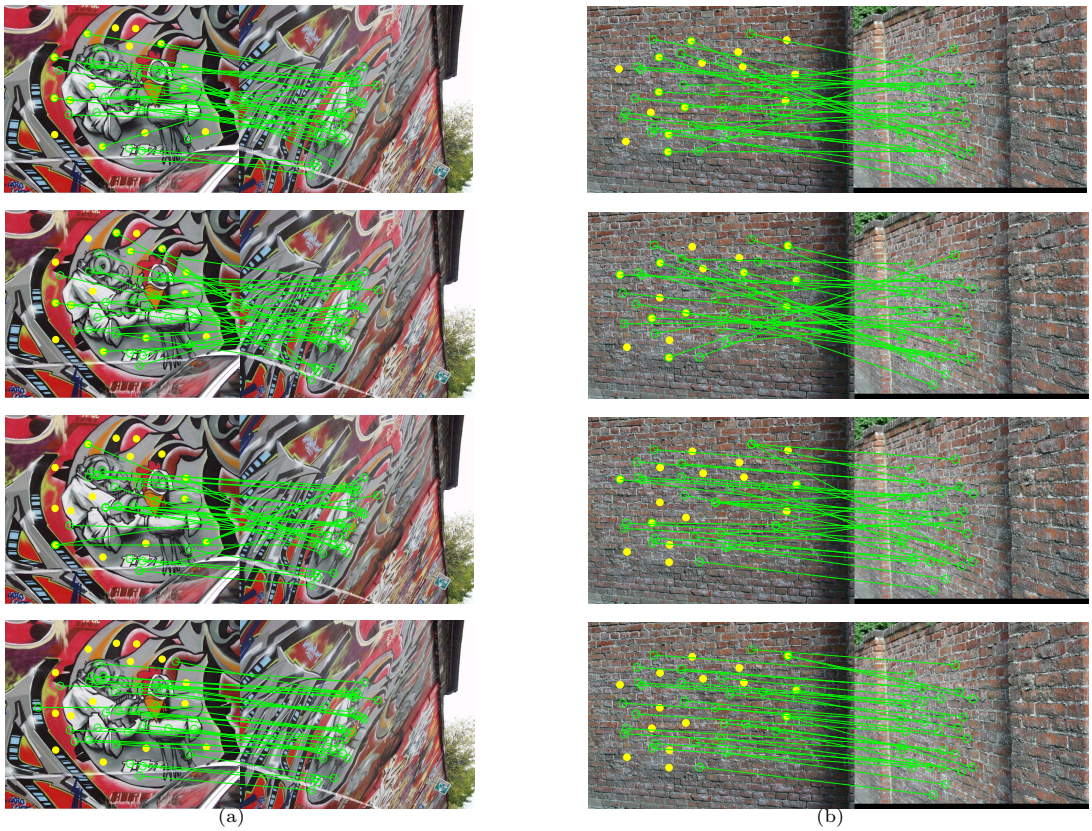


Figure 6: Matching images. Left: `graf` set, right: `wall` set. Top to bottom: results of the spectral method [7], the hypergraph matching method [20], the tensor based method [8] and our method. The feature set P_1 contains 50% outliers.

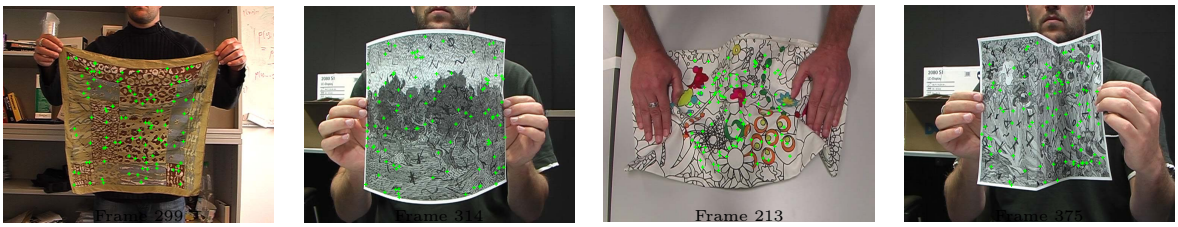


Figure 7: Left to right: point sets on the surface of a piece of cloth, a piece of smoothly curving paper, a creased cushion and a piece of creased paper. The frame number is below each image.

accurate than the other high-order methods [20, 8].

5.4 Image matching under projective transformation

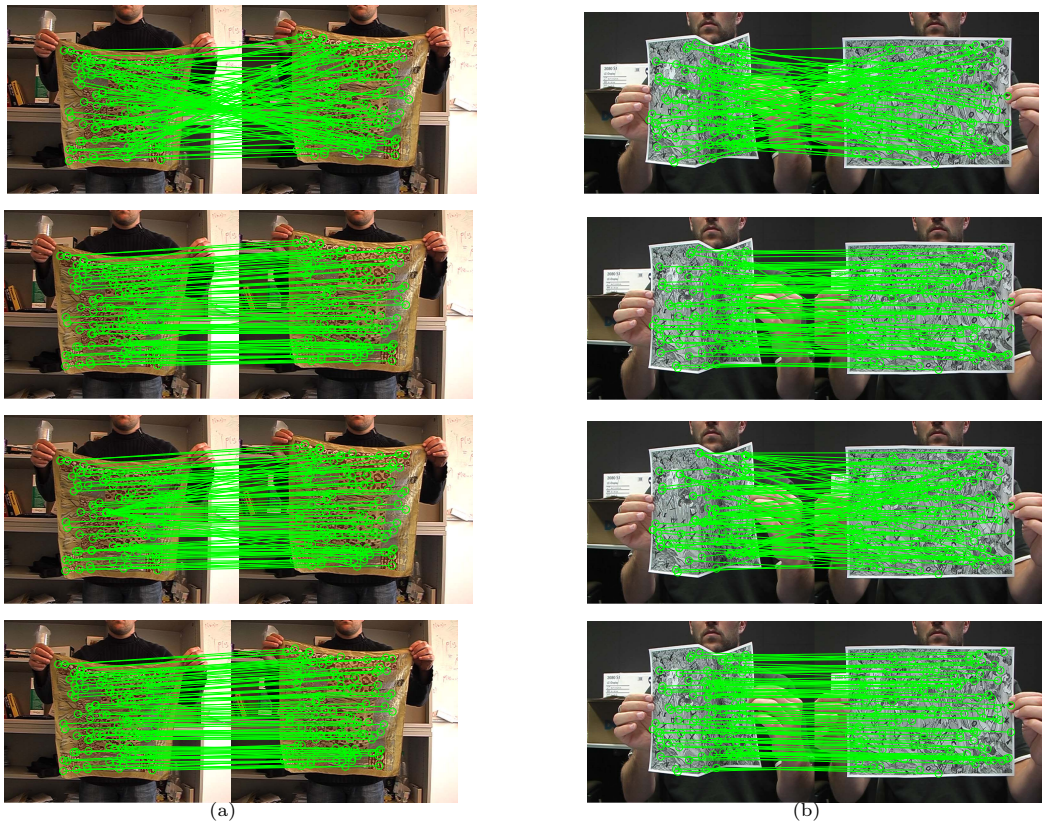
Finally, we used a `house` data set[4] to assess matching performance under perspective transformation. The data include ten different viewpoints under projective transformation, numbered from 0–9. The Harris corners in every frame are provided. We used the Harris corners in image 0 as feature set P_1, and those in other images as P_2. As ground truth correspondences are provided, we can directly compute the matching accuracy of the algorithm, as the number of correctly matched points divided by the total number of points that could potentially be matched. All the higher-order methods used the same third-order potentials from Section **??**, and equivalent tensor sizes.

The larger the baseline separating the images, the larger the relative deformation, and hence the more difficult to match. From Fig. 9, it is clear that our method is remains stable as the baseline increases, and is the most accurate method. Even though there are only seven Harris corners in each frame, it is still difficult to match because of the large viewpoint differential. The average number of sampling feature tuples for all nine baseline matches are 13161, 643305 and 26322 for our method, the method in [8] and the method in [20] respectively.

[4] Fromhttp://www.robots.ox.ac.uk/~vgg/data/data-mview.html

6 Conclusion

This paper has given an efficient higher-order matching algorithm based on the supersymmetric affinity tensor, based on an efficient power iteration method. Our main contributions are as follows. First, we use a higher-order supersymmetric affinity tensor with a compact form to express higher-order consistency constraints of features. Secondly, we derive an efficient higher-order power iteration method, which makes significant savings by taking advantage of supersymmetry. We also give an efficient sampling strategy for choosing feature tuples to create the affinity tensor. Finally, we give a fourth-order potential which possesses affine invariance. Our experiments on both synthetic and real image data sets show that our method has improved matching performance compared to state-of-the-art approaches.



(a) (b)

Figure 8: Matching results. Left: cloth set, frames 258 and299, right: creased paper set, frames 309 and 375. Top to bottom, spectral method [7], hyper graph matching method [20], a tensor based method [8] and our method.

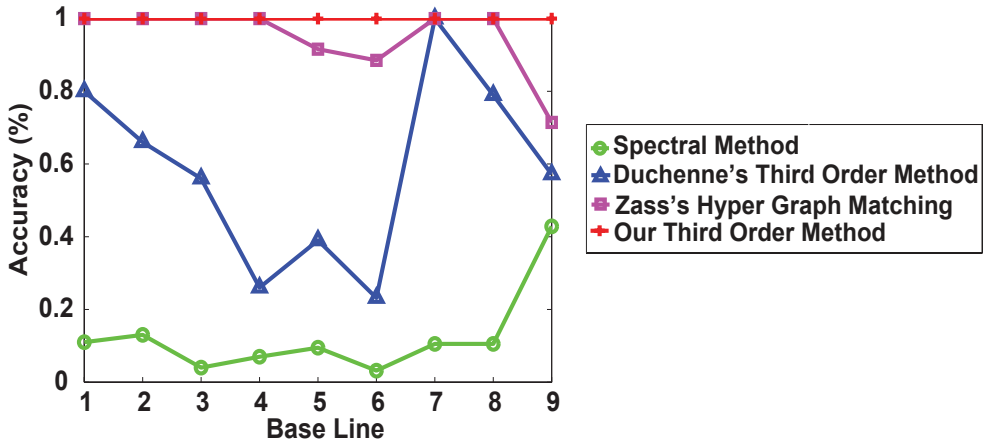


Figure 9: Accuracy as a function of base line.

A Why our Supersymmetric Tensor Based Higher-order Power Iteration Algorithm is efficient?

We explain the idea using a third-order affinity tensor as an example. Given a potential index set θ_3 and a potential function ϕ_3, replacing all the equivalent elements in Equ.(**??**) by a single element, we get the following equations:

$$\forall(i,j,l) \in \theta_3 \ ,$$

$$\begin{aligned} v_i^{(k)} &= \mathcal{T}_3(i,j,l) 2 v_i^{(k-1)} {v_j}^{2(k-1)} {v_l}^{2(k-1)} + \mathcal{T}_3(i,l,j) 2 v_i^{(k-1)} {v_l}^{2(k-1)} {v_j}^{2(k-1)} \\ &= 2 \cdot \mathcal{T}_3(\theta_3(i,j,l))\, 2\, v_i^{(k-1)} {v_j}^{2(k-1)} {v_l}^{2(k-1)} = 2 \cdot \phi_3(i,j,l)\, 2\, v_i^{(k-1)} {v_j}^{2(k-1)} {v_l}^{2(k-1)} \end{aligned} \tag{10}$$

$$\begin{aligned} v_j^{(k)} &= \mathcal{T}_3(j,i,l) 2 v_j^{(k-1)} {v_i}^{2(k-1)} {v_l}^{2(k-1)} + \mathcal{T}_3(j,l,i) 2 v_j^{(k-1)} {v_l}^{2(k-1)} {v_i}^{2(k-1)} \\ &= 2 \cdot \mathcal{T}_3(\theta_3(i,j,l))\, 2\, v_j^{(k-1)} {v_i}^{2(k-1)} {v_l}^{2(k-1)} = 2 \cdot \phi_3(i,j,l)\, 2\, v_j^{(k-1)} {v_i}^{2(k-1)} {v_l}^{2(k-1)} \end{aligned} \tag{11}$$

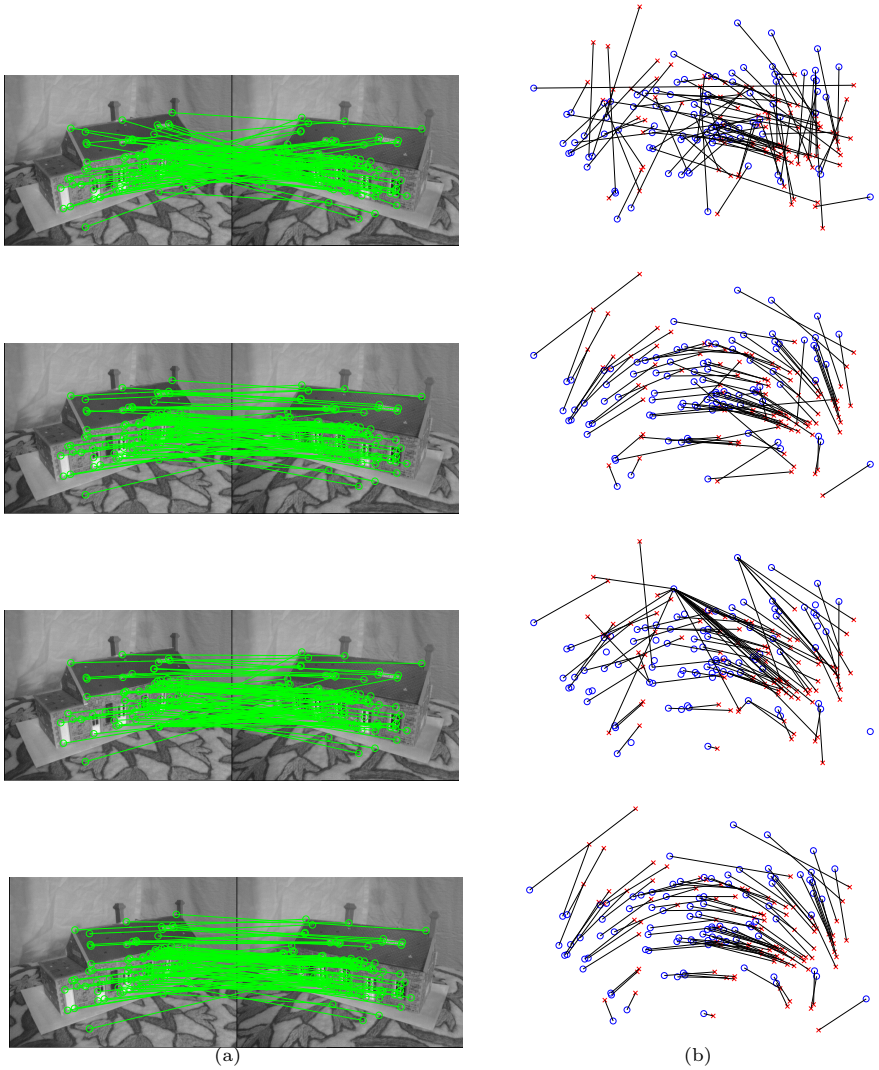
$$\begin{aligned} v_l^{(k)} &= \mathcal{T}_3(l,i,j) 2 v_l^{(k-1)} {v_i}^{2(k-1)} {v_j}^{2(k-1)} + \mathcal{T}_3(l,j,i) 2 v_l^{(k-1)} {v_j}^{2(k-1)} {v_i}^{2(k-1)} \\ &= 2 \cdot \mathcal{T}_3(\theta_3(i,j,l))\, 2\, v_l^{(k-1)} {v_i}^{2(k-1)} {v_j}^{2(k-1)} = 2 \cdot \phi_3(i,j,l)\, 2\, v_l^{(k-1)} {v_i}^{2(k-1)} \end{aligned} \tag{12}$$

References

[1] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust pairwise surface registration. *ACM Transactions on Graphics*, 27(3), 2008.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[3] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 26–33, 2005.



(a) (b)

Figure 10: Matching images 0 and 6 for the `house` data. Top to bottom: results of the spectral method [7], the hyper graph matching method [20], the tensor based method [8] and our method. Left: matches. Right: scatter diagrams of matching results; red × and blue ∘ represent two graphs respectively.

[4] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics*, 30:1:1–1:20, February 2011.

[5] M. Chertok and Y. Keller. Efficient high order matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:2205–2215, 2010.

[6] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[7] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Advanced in Neural Information Processing Systems*, 2006.

[8] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. pages 1980–1987, 2009.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[10] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[11] E. Kofidis and P. A. Regalia. On the best rank-1 approximation of higher-order supersymmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 23(3):863–884, 2002.

[12] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.

[13] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[14] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision*, pages 1482–1489, 2005.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[16] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[17] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 154:1–154:10, 2011.

[18] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proceedings of the Symposium on Geometry Processing*, pages 1383–1392. Eurographics Association, 2009.

[19] A. Wang, S. Li, and L. Zeng. Multiple order graph matching. In *10th Asian Conference on Computer Vision*, pages 471–482, 2010.

[20] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[21] Y. Zheng and D. Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):643–649, 2006.