

# Feature Matching using Supersymmetric Geometric Constraints

## Abstract

Feature matching is a challenging problem lying at the heart of numerous computer graphics and computer vision applications. We present here the *SuperMatching* feature matching algorithm for finding correspondences between two sets of features. *SuperMatching* finds matches of feature points by considering triangle or higher-order polygons formed by such points, going beyond the pointwise and pairwise approaches typically used. It is formulated as a supersymmetric-tensor-based matching scheme, casting feature matching as a higher-order graph matching problem. The supersymmetric tensor represents an affinity metric which takes into account geometric constraints between features. *SuperMatching* exploits a compact form of this affinity tensor, whilst also taking advantage of supersymmetry to devise an efficient sampling strategy to create the affinity tensor. Matching is performed by computing a rank-one approximation of the tensor directly using a higher-order power solution method. Experiments on both synthetic and real scanned data show that our algorithm provides accurate and robust feature matching even in complex cases.

**Keywords:** Feature matching; Geometric constraints; Supersymmetric tensor

## 1 Introduction

Building correspondences between two sets of features of 2D/3D shapes is a fundamental problem in many geometric processing, computer graphics and computer vision tasks. It arises in applications such as feature extraction [Johnson and Hebert 1999; Lowe 2004; Sun et al. 2009; Toler-Franklin et al. 2010; Leutenegger et al. 2011], shape matching [Belongie et al. 2002; Berg et al. 2005; Brown and Rusinkiewicz 2007; Tevs et al. 2009; Ovsjanikov et al. 2010; Tevs et al. 2011; Sahillioglu and Yemez 2011; Windheuser et al. 2011], registration of 3D shapes [Gelfand et al. 2005; Aiger et al. 2008; Li et al. 2008; Zeng et al. 2010; van Kaick et al. 2011; Chang and Zwicker 2011], automatic shape understanding [Lipman and Funkhouser 2009; Sun et al. 2010; Kim et al. 2011], shape retrieval from database [Bronstein et al. 2011], and reconstruction [Brown and Rusinkiewicz 2007; Chang and Zwicker 2011].

In principle and practise, the correspondence building process mainly proceed in three steps [Lowe 2004; Leutenegger et al. 2011]: salient feature detection, high-quality descriptor, and accurately matching. The former two problems have been widely attracted considerable attention as their importance is easy perceivable. However, even with the ideal feature detector and descriptor that capturing the most important and distinctive information content enclosed in the detected salient regions, the correspondences still could not ideally built in the state-of-the-art algorithms [van Kaick et al. 2011]. The reason is that the real input data is so imperfect and complex especially with symmetric and repetitive regions. The researchers are beginning to be aware of that the limitation could be effectively alleviated by feature matching. Some original idea (e.g. RANSAC-like algorithms [Tevs et al. 2009; Tevs et al. 2011]) has been extended, or indirectly sorting to shape embedding strategies, such as Generalized Multidimensional Scaling [Bronstein et al. 2011], Heat Kernel Map [Ovsjanikov et al. 2010], Möbius Transformation [Lipman and Funkhouser 2009; Kim et al.

2011]. But, these previous algorithms still did not treat feature matching as one independent problem, although they agreed with that matching is not tightly coupled with feature detection and description.

In the paper, we mainly focus on feature matching problem, which is complementary to existing correspondence algorithms. As we know, a matching itself may comprise one or more items of a given kind. We may match single point to single point (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. As pointed by [Conte et al. 2004], when single features are matched, we must solve a linear assignment problem, if multiple features are matched at once, a quadratic or higher-order assignment problem results.

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 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, Leordeanu and Hebert [Leordeanu and Hebert 2005] 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. The idea has been widely adopted in the 3D shape matching algorithms [Tevs et al. 2009; Ovsjanikov et al. 2010; Tevs et al. 2011; Kim et al. 2011; Sahillioglu and Yemez 2011; Windheuser et al. 2011].

Higher-order assignment further generalizes the assignment problem to include yet more complex constraints between features. For example, third-order potential functions, proposed in [Duchenne et al. 2009; Zeng et al. 2010; Chertok and Keller 2010], quantify 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 [Kofidis and Regalia 2002], the unaccurate limitation could be further solved by our algorithm.

Thus, by formulating the higher-order problem using a supersymmetric affinity tensor, we propose an accurate and efficient higher-order supersymmetric matching algorithm, named *SuperMatching*. *SuperMatching*, addressed in Section 4, works well when matching a moderate number of features using triples or higher polygons of features. The contributions of this paper include:

- We show how to define a compact higher-order supersymmet-

ric affinity tensor to express geometric consistency constraints between tuples of features.

- 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 in Section ?? on both synthetic and real scanned data sets show that SuperMatching result is accurate and robust, and has competitive computational cost compared to previous algorithms.

## 2 Related work

According to the applied constraints, previous approaches to feature matching can be classified into those 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 is a linear assignment problem which only considers an affinity measure between two graph nodes, one from each set being matched; this measure is typically the feature distance between the two feature points. In concrete terms, the linear assignment problem may be expressed as: find a mapping  $f: P_1 \rightarrow P_2$ . Affinity measures used in computer vision and computer graphics tasks rely heavily on descriptors computed using local information around each feature point, e.g. SIFT [Lowe 2004], spin images [Johnson and Hebert 1999], heat diffusion signatures [Sun et al. 2009], and BRISK [Leutenegger et al. 2011]. It is apparent that point-to-point matching is weak in that wrong correspondences may be readily established.

Matching pairs of points in one set to pairs of points in the other set leads to a quadratic assignment problem. The usual approach is now to take into account both similarity of the point features and the Euclidean/Geodesic distance between the points in a pair, assuming the objects are related by a rigid body transform [Leordeanu and Hebert 2005], or at least an isometry [Li et al. 2008; Tevs et al. 2009; Ovsjanikov et al. 2010; Tevs et al. 2011; Sahillioglu and Yemez 2011; Windheuser et al. 2011]. The quadratic assignment problem seeks to find a mapping which represents the optimal assignment. Unfortunately, this problem is NP-hard, unreasonable matching results could not be avoided.

Several higher-order approaches have also been proposed. Such higher-order methods can significantly improve matching accuracy, but higher-order assignment problem is again NP-hard, and various approximate methods have again been developed. Zass and Sashua [Zass and Sashua 2008] 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. [Duchenne et al. 2009] 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. [Chertok and Keller 2010] 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. [Wang et al. 2010] also build a third-order affinity tensor, and obtain 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 which do not accurately meet the assumption that the datasets are related by an isometry, the matching results may become unstable.

A related idea using higher constraints in 3D registration, 4-points congruent sets method (4PCS), was proposed by Aiger et al. [Aiger et al. 2008]. It is a fast alignment scheme for 3D point sets that uses widely separated points. However, the coplanar 4 points and rigid-transformation constraints are some strong, and limit its application in real data. We solve both rigid and non-rigid matching problem by one more mathematical and formulated tensor-based algorithm.

## 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  $\mathbf{x} \in \{0, 1\}^{N_1 N_2}$ .

Solving the higher-order matching problem is equivalent to find the optimal assignment vector  $\mathbf{x}^* = \langle x_{i_1}, \dots, x_{i_N} \rangle \in \{0, 1\}^{N_1 N_2}$ , satisfying

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

Here,  $i_n$  stands for an assignment  $(i'_n, j'_n)$ , ( $i'_n \in P_1, j'_n \in P_2$ ), and the product  $x_{i_1} \dots x_{i_N}$  is 1 only if all assignments  $\{i_n\}_{n=1}^N$  are equal to 1, which means the tuple of features  $(i'_1, \dots, i'_N)$  from  $P_1$  is matched correspondingly to the tuple of features  $(j'_1, \dots, j'_N)$  from  $P_2$ .  $\mathcal{T}_N(i_1, \dots, i_N)$  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  $(i'_1, \dots, i'_N)$  and  $(j'_1, \dots, j'_N)$ .

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  $\phi_N$ ? (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 Features 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 [Kolda and Bader 2009]. 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 [Kofidis and Regalia 2002].*

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  $T_N(i_1, \dots, i_N)$ , in Equ.(1), which measures the affinity of the assignments  $(i_1, \dots, i_N)$ ; in other words it uses the value  $\phi_N(i_1, \dots, i_N)$  to give a score when matching the ordered feature tuple  $(i'_1, \dots, i'_N)$  from  $P_1$  to the ordered feature tuple  $(j'_1, \dots, j'_N)$  from  $P_2$ . Often, the value of  $\phi_N(i_1, \dots, i_N)$  is invariant under permutation of its arguments  $(i_1, \dots, i_N)$ . 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  $N^{\text{th}}$  order  $I_1 \times \dots \times I_N$ , nonnegative tensor  $\mathcal{T}_N$ , where  $I_1 = I_2 = \dots = I_N = \{1, \dots, N_1 N_2\}$ , for which there exists a set of indices  $\theta_N$ , and an  $N^{\text{th}}$  order potential function  $\phi_N$ , such that*

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

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

A tensor element with  $(i_1, i_2, \dots, i_N) \in \theta_N$  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  $\mathcal{T}_N(i_1, i_2, \dots, i_N)$  by the canonical entry  $\mathcal{T}_N(\text{sort}(i_1, i_2, \dots, i_N))$ ,  $\forall (i_1, i_2, \dots, i_N) \in \theta_N$ . 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 Higher-order Power Iteration Solving

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

$$\begin{aligned} \mathbf{x}^* &= \arg \max_{\mathbf{x}} \sum_{i_1, i_2, \dots, i_N} \mathcal{T}_N(i_1, \dots, i_N) x_{i_1} \dots x_{i_N} \\ &= \max \langle \mathcal{T}_N, \mathbf{x}^{*N} \rangle \end{aligned} \quad (3)$$

where  $\star$  is called the Tucker product [Kofidis and Regalia 2002], and  $\mathbf{x} \in \{0, 1\}^{N_1 N_2}$ .

As noted, solving Equ.(3) is an NP-complete problem, so it is common to relax the constraints: the binary assignment vector  $\mathbf{x} \in \{0, 1\}^{N_1 N_2}$  is replaced by an assignment vector  $\mathbf{u}$  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  $\mathcal{T}_N$  [Lathauwer et al. 2000], i.e. finding a scalar  $\lambda$  and a unit norm vector  $\mathbf{u} \in \mathbb{R}^{N_1 N_2}$ , such that the tensor  $\hat{\mathcal{T}}_N = \lambda \mathbf{u} \star \mathbf{u} \star \dots \star \mathbf{u} = \mathbf{u}^{*N}$  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  $\mathbf{u}$  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 [Kofidis and Regalia 2002]. The S-HOPM algorithm converges under the assumption of convexity for the functional induced by the tensor [Kofidis and Regalia 2002], which is satisfied in many practical applications.

In their recent work, Duchenne et al. [Duchenne et al. 2009] failed to note that the whole iteration process can be simplified by taking advantage of supersymmetry. In [Duchenne et al. 2009], all tensor elements take part in the iteration process, which is unnecessary. 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 [Duchenne et al. 2009] 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.

We solve the above problems by proposing a new efficient higher-order power iteration algorithm, Algorithm 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  $N$ th-order supersymmetric affinity tensor, the corresponding result is:

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

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  $\mathcal{T}_n(i_1, i_2, \dots, i_n)$ , which are expressed by a single potential function  $\phi_n(i_1, i_2, \dots, i_n)$ . 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 [Kofidis and Regalia 2002]. We simply use positive random values to initialize  $u_0$ , which ensures that the algorithm converges.

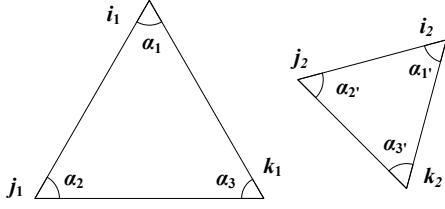


Figure 1: Third-order potential (triangle).

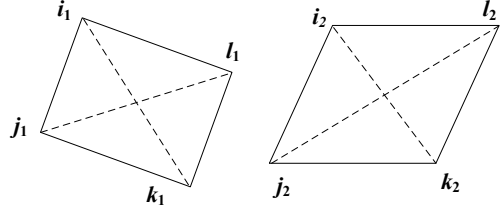


Figure 2: Fourth-order potential (quadrangle).

**Algorithm 1** Higher-order power iteration method for a supersymmetric affinity tensor (with  $C_1$  norm)

**Input:**  $N$ th-order supersymmetric affinity tensor  $\mathcal{T}_n$

**Output:** unit-norm vector  $\mathbf{u}$

```

1: Initialize  $\mathbf{u}_0, k = 1$ 
2: repeat
3:   for all  $(i_1, i_2, \dots, i_N) \in \theta_N$  do
4:     for all  $m \in (i_1, i_2, \dots, i_N)$  do
5:        $v_m^{(k)} = \frac{(N-1)! \cdot \phi_N(i_1, i_2, \dots, i_N)}{2v_m^{(k-1)}v_{i_1}^{(k-1)} \dots v_{m-1}^{(k-1)}v_{m+1}^{(k-1)} \dots v_{i_N}^{(k-1)}}$ 
6:     end
7:     for  $i = 1 : N_1$  do
8:        $v_i^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2) = \hat{v}^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2) / \|\hat{v}^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2)\|_2$ 
9:     end
10:  end
11:   $k = k + 1$ 
12: until convergence;
    Note:  $v_i^{(k)}(((i-1) \cdot N_2 + 1) : i \cdot N_2)$  denotes the slice of  $v^{(k)}$  with indices from  $(i-1) \cdot N_2 + 1$  to  $i \cdot N_2$ .
```

### 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 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 [Duchenne et al. 2009; Chertok and Keller 2010] third-order geometric-similarity invariant potential  $\phi_3$  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  $\phi_3$  can be defined in terms of differences of corresponding interior angles:

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

where  $\varepsilon > 0$  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  $\phi_4$  which is affine-invariant, linking feature tuples with four features each. We use affine invariance of the ratio between

two closed areas to define  $\phi_4$  as:

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

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  $N$ th-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  $\forall (f_{i_1}^1, f_{i_2}^1, \dots, f_{i_N}^1) \in F_1$ , calculating the potentials for  $(f_{i_1}^1, f_{i_2}^1, \dots, f_{i_N}^1)$  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 this 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,  $\forall (f_{i_1}^1, f_{i_2}^1, \dots, f_{i_N}^1) \in F_1$ , we find the  $k$  most similar features in  $F_2$  to build  $k$  potential elements as  $\phi_i^k$ . 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:

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

where  $\Omega$  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

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

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

and  $\Omega$  is arbitrary permutation. This sampling constraint eliminates overlaps that may appear in the potential elements  $\theta_N$ .

When  $N_1$  is not equal to  $N_2$  (suppose  $N_1 < N_2$ ), we perform sampling as in [Duchenne et al. 2009], 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 [Duchenne et al. 2009]. 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 [Duchenne et al. 2009; Zass and Shashua 2008] 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 Conclusion

a novel matching algorithm named SuperMatching, which tackles the classic Computer Graphics and Computer Vision problem of matching various features for cases without any assumptions.

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 Why higher-order power iteration solving of supersymmetric tensor is efficient?

We explain the idea using a third-order affinity tensor as an example. Given a potential index set  $\theta_3$  and a potential function  $\phi_3$ , re-

placing all the equivalent elements in Equ.(?) by a single element, we get the following equations:

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

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

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

## References

- AIGER, D., MITRA, N. J., AND COHEN-OR, D. 2008. 4-points congruent sets for robust pairwise surface registration. *ACM Transactions on Graphics* 27, 3.
- BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 4, 509–522.
- BERG, A. C., BERG, T. L., AND MALIK, J. 2005. Shape matching and object recognition using low distortion correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, 26–33.
- BRONSTEIN, A. M., BRONSTEIN, M. M., GUIBAS, L. J., AND OVSJANIKOV, M. 2011. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics* 30, 1:1–1:20.
- BROWN, B. J., AND RUSINKIEWICZ, S. 2007. Global non-rigid alignment of 3-d scans. *ACM Transactions on Graphics (SIGGRAPH)* 26.
- CHANG, W., AND ZWICKER, M. 2011. Global registration of dynamic range scans for articulated model reconstruction. *ACM Transactions on Graphics* 30, 26:1–26:15.
- CHERTOK, M., AND KELLER, Y. 2010. Efficient high order matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 2205–2215.
- CONTE, D., FOGGIA, P., SANSONE, C., AND VENTO, M. 2004. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 18, 3, 265–298.
- DUCHENNE, O., BACH, F., KWEON, I., AND PONCE, J. 2009. A tensor-based algorithm for high-order graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1980–1987.
- GELFAND, N., MITRA, N. J., GUIBAS, L. J., AND POTTSMANN, H. 2005. Robust global registration. In *Proceedings of the third Eurographics symposium on Geometry processing*.
- JOHNSON, A. E., AND HEBERT, M. 1999. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 5, 433–449.

- 468 KIM, V. G., LIPMAN, Y., AND FUNKHOUSER, T. 2011. Blended  
469 intrinsic maps. In *SIGGRAPH*, 79:1–79:12.
- 470 KOFIDIS, E., AND REGALIA, P. A. 2002. On the best rank-1  
471 approximation of higher-order supersymmetric tensors. *SIAM*  
472 *Journal on Matrix Analysis and Applications* 23, 3, 863–884.
- 473 KOLDA, T. G., AND BADER, B. W. 2009. Tensor decompositions  
474 and applications. *SIAM Review* 51, 3 (September), 455–500.
- 475 LATHAUWER, L. D., MOOR, B. D., AND VANDEWALLE, J. 2000.  
476 On the best rank-1 and rank-( $r_1, r_2, \dots, r_n$ ) approximation of  
477 higher-order tensors. *SIAM Journal on Matrix Analysis and Ap-*  
478 *plications* 21, 4, 1324–1342.
- 479 LEORDEANU, M., AND HEBERT, M. 2005. A spectral technique  
480 for correspondence problems using pairwise constraints. In *In-*  
481 *ternational Conference of Computer Vision*, 1482–1489.
- 482 LEUTENEGGER, S., CHLI, M., AND SIEGWART, R. 2011. Brisk:  
483 Binary robust invariant scalable keypoints. In *International Con-*  
484 *ference of Computer Vision*.
- 485 LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global corre-  
486 spondence optimization for non-rigid registration of depth scans.  
487 *Computer Graphics Forum (Proc. SGP)* 27, 5, 1421–1430.
- 488 LIPMAN, Y., AND FUNKHOUSER, T. 2009. Möbius voting for  
489 surface correspondence. *ACM Transactions on Graphics (SIG-*  
490 *GRAPH)* 28, 72:1–72:12.
- 491 LOWE, D. G. 2004. Distinctive image features from scale-invariant  
492 keypoints. *International Journal of Computer Vision* 60, 2, 91–  
493 110.
- 494 OVSIJANIKOV, M., MRIGOT, Q., MMOLI, F., AND GUIBAS, L.  
495 2010. One point isometric matching with the heat kernel. *Com-*  
496 *puter Graphics Forum (SGP)* 29, 5, 1555–1564.
- 497 SAHILLIOGLU, Y., AND YEMEZ, Y. 2011. Coarse-to-fine com-  
498 binatorial matching for dense isometric shape correspondence.  
499 *Computer Graphics Forum (SGP)* 30, 5, 1461–1470.
- 500 SUN, J., OVSIJANIKOV, M., AND GUIBAS, L. 2009. A concise and  
501 provably informative multi-scale signature based on heat diffu-  
502 sion. In *Proceedings of the Symposium on Geometry Processing*,  
503 Eurographics Association, 1383–1392.
- 504 SUN, J., CHEN, X., AND FUNKHOUSER, T. A. 2010. Fuzzy  
505 geodesics and consistent sparse correspondences for deformable  
506 shapes. *Computer Graphics Forum* 29, 5, 1535–1544.
- 507 TEVS, A., BOKELOH, M., WAND, M., SCHILLING, A., AND SEI-  
508 DEL, H.-P. 2009. Isometric registration of ambiguous and partial  
509 data. In *CVPR*, 1185–1192.
- 510 TEVS, A., BERNER, A., WAND, M., IHRKE, I., AND SEIDEL,  
511 H.-P. 2011. Intrinsic shape matching by planned landmark sam-  
512 pling. *Computer Graphics Forum* 30, 2, 543–552.
- 513 TOLER-FRANKLIN, C., BROWN, B., WEYRICH, T.,  
514 FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2010. Multi-  
515 feature matching of fresco fragments. *ACM Transactions on*  
516 *Graphics (SIGGRAPH ASIA)* 29, 185:1–185:12.
- 517 VAN KAICK, O., ZHANG, H., HAMARNEH, G., AND COHEN-  
518 OR, D. 2011. A survey on shape correspondence. *Computer*  
519 *Graphics Forum* 30, 6.
- 520 WANG, A., LI, S., AND ZENG, L. 2010. Multiple order graph  
521 matching. In *10th Asian Conference on Computer Vision*, 471–  
522 482.
- 523 WINDHEUSER, T., SCHLICKWEI, U., SCHIMDT, F. R., AND  
524 CREMERS, D. 2011. Large-scale integer linear programming  
525 for orientation preserving 3d shape matching. *Computer Graph-*  
526 *ics Forum (SGP)* 30, 5, 1471–1480.
- 527 ZASS, R., AND SHASHUA, A. 2008. Probabilistic graph and hy-  
528 pergraph matching. In *IEEE Conference on Computer Vision*  
529 *and Pattern Recognition*, 1–8.
- 530 ZENG, Y., WANG, C., WANG, Y., GU, X., SAMARAS, D., AND  
531 PARAGIOS, N. 2010. Dense non-rigid surface registration using  
532 high-order graph matching. In *CVPR*, 382–389.