SuperMatching: Feature Matching using Supersymmetric Geometric Constraints

Zhi-Quan Cheng, Yin Chen, Ralph R. Martin, Yu-Kun Lai, Aiping Wang

Abstract—Feature matching is a challenging problem at the heart of numerous computer graphics and computer vision applications. We present the *SuperMatching* algorithm for finding correspondences between two sets of features. It does so by considering triples or higher-order tuples of points, going beyond the pointwise and pairwise approaches typically used. SuperMatching is formulated using a supersymmetric tensor representing an affinity metric which takes into account feature similarity and geometric constraints between features: feature matching is cast as a higher-order graph matching problem. SuperMatching takes advantage of supersymmetry to devise an efficient sampling strategy to estimate the affinity tensor, as well as to store the estimated tensor compactly. Matching is performed by an efficient higher-order power iteration approach which takes advantage of this compact representation. Experiments on both synthetic and real data show that SuperMatching provides more accurate feature matching than other state-of-the-art approaches for a wide range of 2D and 3D features, with competitive computational cost.

Index Terms—Feature matching, Geometric constraints, Supersymmetric tensor.

1 Introduction

Building correspondences between two sets of features belonging to a pair of 2D images or 3D shapes is a fundamental problem in many computer graphics, geometry processing, and computer vision tasks. It arises in applications such as registration of partial or entire 3D shapes [1]–[2], shape retrieval from databases [3], shape matching [4]–[5], shape reconstruction [2], [6], [7], [8], and automatic shape understanding [9]–[10].

Correspondence determination is typically done in three steps [11]–[12]: (i) computing high-quality descriptors which serve to distinguish points from one another, (ii) choosing certain salient points with unusual feature descriptors, for matching, and (iii) determining the most suitable matching between the two sets of points. The former two problems have attracted considerable attention; their importance is clear. However, even supposing ideal feature descriptors and selectors that capture the most important and distinctive information about the neighborhood of each salient point, state-of-the-art algorithms still find it challenging to determine the best matching [13]. Real input data is noisy, and data may only be approximately in correspondence; the problem is further complicated by the presence of symmetric and congruent regions. Various feature matching algorithms have been devised to be robust in the presence of such issues. RANSAC-like algorithms [14], [15] minimize the effects of outliers, while generalized multidimensional scaling [3] and heat kernel maps [16] consider the manifold in which the points are embedded. Möbius transformations [17], [10] also provide a powerful approach. However, this previous work generally does not treat the matching step as a separate task, even if matching is not tightly coupled with feature description and selection. This paper focuses on the feature matching problem, treating it as an independent problem in its own right.

Matching may be done pointwise (single points to single points), or using tuples of points: e.g. point pairs, separated by a fixed distance, to other point pairs, triples of points forming a triangle to other triples of points, and so on. As pointed out by [18], matching single features leads to a linear assignment problem, but if multiple features are matched simultaneously, a quadratic or higher-order assignment problem results. 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, or similar local appearance. Quadratic and higher-order assignment matches groups of features, enforcing other constraints such as consistency of distances between the points in each tuple being matched. Doing so helps to reject many false matches, greatly improving matching output. Feature similarity and satisfaction of constraints may in general be expressed in terms of an affinity tensor relating pairs of point tuples.

As a particular example of *quadratic* assignment, Leordanu and Hebert [19] consider pairs of feature descriptors, and use *distances* between pairs of features

- *Zhi-Quan Cheng, Yin Chen, and Aiping Wang are with the National Laboratory for Parallel and Distributed Processing, School of Computing, National University of Defense Technology, Changsha, Hunan Province, China, 410073.*
  E-mail: see http://www.computer-graphics.cn
- *Ralph R. Martin and Yu-Kun Lai are with School of Computer Science & Informatics, Cardiff University, Cardiff, Wales, UK, CF24 3AA.*

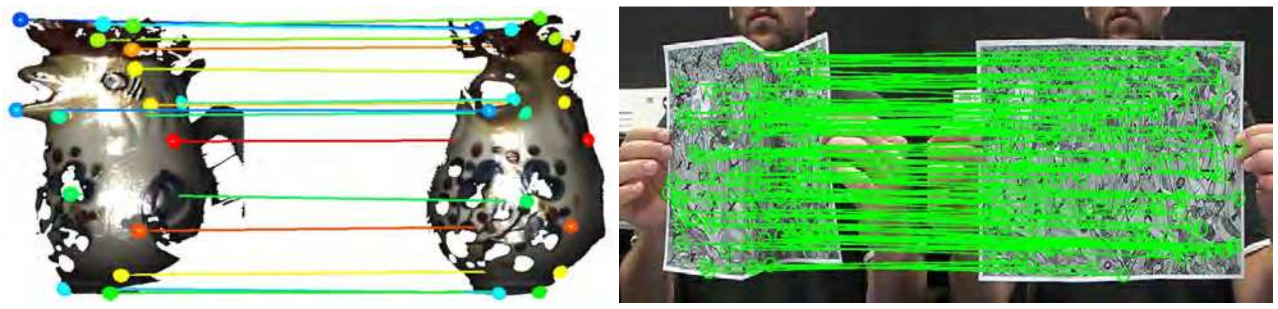


Fig. 1. Correspondences between datasets determined by SuperMatching. Feature points were created by (left example) simple uniform sampling of rigid scans, (right example) SIFT, on an isometrically deforming surface. For clarity, only a few representative matches are shown.

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 3D shape matching algorithms [14]–[5], [10].

Higher-order assignment includes yet more complex constraints between features. For example, third-order potential functions, used in [20], [21]–[22], quantify the affinity between two point triples by measuring the similarity of the angles of the triangles formed by such triples. Note that this angular similarity value only considers the *total* difference in corresponding angles, and does not change with reordering of elements in the tuple. When similarity is expressed in this way, the affinity tensor becomes a *supersymmetric* tensor [23], whose entries remain unchanged under any permutation of its indices. As shown in [23], such rich symmetry of the tensor permits a simplified version of the solution to be applicable, under assumptions of convexity (or concavity) for the functional induced by the tensor in question. This in turn provides significant savings in computational time as compared to the unconstrained solution method.

Going beyond the supersymmetric form in [23], our SuperMatching algorithm also formulates higher-order matching problems using a supersymmetric affinity tensor scheme. A new supersymmetric affinity tensor definition is introduced, and we use it to deduce a more compact expression than in previous work. This allows us to devise a compact higher-order power iteration solution for the higher-order matching problem. SuperMatching can accurately match a moderate number (several hundreds) of features using triples or larger tuples of features. The contributions of this paper include:

- We show how to define a compact higher-order supersymmetric affinity tensor to express geometrically consistent constraints between feature tuples.
- Complete computation of the full affinity tensor is computationally infeasible. We efficiently estimate it using a sampling strategy which takes advantage of supersymmetry. This avoids sampling repetitive items, it allows the tensor to be stored compactly, and also improves the matching accuracy by avoiding imbalances in sampling.
- We make full use of the compactness of the affinity tensor to deduce a power iteration method which efficiently solves the matching problem.

Our experiments using both synthetic and real captured data sets show that SuperMatching is more accurate and robust than prior methods, yet has similar computational cost. Importantly, it is a general matching approach, independent of choice of 2D or 3D feature descriptors.

2 Related work

Previous approaches to feature matching can be classified as those which match single points to single points, those which match point pairs to point pairs, and so on.

Matching single points to single points leads to a linear assignment problem which only considers an affinity measure between two individual features, one from each set being matched. The affinity measure is typically defined as the feature distance between the feature vectors, which in turn are based on local information around each feature point, e.g. SIFT [24], spin images [11], heat diffusion signatures [25], and BRISK [12]. Point-to-point matching can give misleading results as wrong correspondences are readily established.

Matching point pairs in one set to point pairs in the other set leads to a quadratic assignment problem. Such methods now take into account both similarity of the point features, *and* either the Euclidean distance between the points in a pair, assuming the two sets of points are related by a rigid transformation [19], [26], or the geodesic distance, assuming isometry [27], [14]–[5]. Unfortunately, this quadratic assignment problem is NP-hard, and again, matches found are not always reliable.

Several higher-order approaches have also been proposed. While they can significantly improve matching accuracy, higher-order assignment is even more computationally demanding, so various approximate methods have been developed. [28] considered 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. [21], [29] introduced the use of a third-order tensor in place of an affinity matrix to represent affinities of feature triples, and higher-order power iteration is used to achieve the final matching. [22] built a unified multiple higher-order affinity tensor, by extending the third-order tensor method [21], [29]. An alternative [30] is to 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. 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 match moderate numbers (hundreds) of features. However, these approaches do not really take proper advantage of supersymmetry of the affinity tensor. SuperMatching does so, leading to an improvement in matching accuracy.

A related idea also using higher order constraints for 3D registration is the 4-points congruent sets method (4PCS) proposed in [31]. It is a fast alignment scheme for 3D point sets that uses widely separated points. However, the need to find coplanar 4-tuples of points and the assumption of their relation by rigid transformation limit its applicability. Our approach can not only be applied to rigid registration, but also to more difficult correspondence issues involving e.g. articulated or isometrically deformed data.

3 Overview

A tensor generalizes 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 [32]. More formally, an N^{th}-order tensor is an element of the tensor product of N vector spaces, each with its own coordinate system. Here, we mainly consider the third-order case, which represents a reasonable trade-off between quality of solution and computational cost, which increases rapidly with tensor order. Many of the ideas generalize to higher order.

Assume we are given two sets of feature points P_1 and P_2, with N_1 and N_2 points respectively. Let $i_n = (f^1_{i_n}, f^2_{i_n})$ be a pair of points from P_1 and P_2, respectively. Matching between these two feature sets can be represented by an *assignment variable* $\boldsymbol{x}$ which is a vector $\in \{0,1\}^{N_1 N_2}$, with each element representing whether a pair $i_n(f^1_{i_n}, f^2_{i_n})$ is selected in the matching (if $x_{i_n} = 1$) or not (if $x_{i_n} = 0$). From the N^{th}-order tensor viewpoint, the matching problem is equivalent to finding the optimal assignment tensor $\boldsymbol{x}^* \in \{0,1\}^{N_1 N_2}$, satisfying [32]

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

Here, $i_n \in \{i_1,\cdots,i_N\}$ stands for an assignment in the n^{th} dimension of the N vector spaces. Let all feature tuples for P_1 and P_2 be F_1 and F_2, then $\forall(f^1_{i_1},\cdots,f^1_{i_N}) \in F_1$, there is a matching to corresponding feature tuples in F_2. For example, given a third-order tensor, $i_n \in \{1,2,3\}$, each index could be expressed as $i_1 = (f^1_{i_1}, f^2_{i_1}), i_2 = (f^1_{i_2}, f^2_{i_2}), i_3 = (f^1_{i_3}, f^2_{i_3})$: pairs of potentially matched points. The product $x_{i_1}\cdots x_{i_N}$ will be equal to 1 if the points $(f^1_{i_1},\cdots,f^1_{i_N})$ are matched to the points $(f^2_{i_1},\cdots,f^2_{i_N})$, and 0 otherwise. $\mathcal{T}_N(i_1,\cdots,i_N)$ is the affinity of the set of assignments $\{i_n\}^N_{n=1}$, which is high if the features in tuple $(f^1_{i_1},\cdots,f^1_{i_N})$ have similar descriptor values to the features in the tuple $(f^2_{i_1},\cdots,f^2_{i_N})$, and have similar distances between them. Note that the size of $\mathcal{T}_N(i_1,\cdots,i_N)$ is $(N_1N_2)^N$. In this paper, the affinity measures expressing similarity of feature tuples are compactly represented and efficiently computed by using the supersymmetric tensor.

In the rest of the paper, we consider the one-to-many correspondence problem. We assume that each point in P_1 is matched to exactly one point in P_2, but that the reverse is not necessarily true. If we *do* 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. This pair of one-to-many matching results can be used as needed in particular applications; most obviously, we could intersect the result sets to build a set of one-to-one correspondences.

From Eq. (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 $\mathcal{T}_N$ in a supersymmetric manner? (see Section 4.1)
- (approximately) solve the optimal higher-order assignment problem efficiently? (see Section 4.2)
- determine an appropriate sampling strategy to estimate the affinity tensor in a way which will give good matching accuracy—it is too large to compute fully? (see Section 4.3)
- define a suitable affinity measure between two feature tuples? (see Section 4.4)

4 SuperMatching

We start by discussing the first two issues mentioned above, which are independent of application; later we turn to sampling strategy and definition of affinity measure, which are application dependent.

4.1 Supersymmetric Affinity Tensor

The supersymmetric higher-order affinity tensor is invariant under permutation of indices. The main motivation of using supersymmetry is to allow us to avoid redundant storage and computation.

Definition 1 (Supersymmetric Tensor): A tensor is *supersymmetric* if its entries are invariant under any permutation of its indices [23].

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

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 $I_1,\cdots,I_N$, nonnegative tensor $\mathcal{T}_N$, 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. θ_N satisfies $i_m \neq i_n$ $\forall(i_1,\ldots,i_N)\in\theta_N, \forall i_m \in \{i_1,\ldots,i_N\}$ and $\forall i_n \in \{i_1,\ldots,i_N\} - \{i_m\}$.

A tensor element with $(i_1,\ldots,i_N)\in\theta_N$ is called a *potential element*, while other elements are called *zero elements*. A potential element represents one matching result out of all possible matching candidates. Potential elements are further detailed in Section 4.3.

Using our new Definition 2, we can reduce the amount of storage needed, by representing every potential element $\mathcal{T}_N(i_1,\ldots,i_N)$ by its canonical entry $\mathcal{T}_N(\mathrm{sort}(i_1,\ldots,i_N))$, $\forall(i_1,\ldots,i_N)\in\theta_N$. Each stored value thus provides the value for $N!$ entries. Furthermore, as zero elements are always zero, there is no need to store them. This greatly reduces both storage, and the amount of feature tuple sampling needed when estimating the affinity tensor, as discussed in Section 4.3. At the same time, it can be used to make the power iteration process more efficient: see Section 4.2.

Algorithm 1 Higher-order power iteration solution (with ℓ^1 norm) for the supersymmetric affinity tensor

Input: N^{th}-order supersymmetric affinity tensor
Output: Unit ℓ^1-norm vector $\boldsymbol{u}$

```
 1: Initialize $\boldsymbol{v}_0$ to random values in [0,1], $k = 1$
 2: repeat
 3:     for all $(i_1,\cdots,i_N)\in\theta_N$ do
 4:         for all $m\in(i_1,\cdots,i_N)$ do
 5:             $v_m^{(k)} = (N-1)!\phi_N(i_1,\cdots,i_N)2v_m^{(k-1)}v_{i_1}^{2(k-1)}\cdots v_{m-1}^{2(k-1)}v_{m+1}^{2(k-1)}\cdots v_{i_N}^{2(k-1)}$
 6:         end
 7:         for $i = 1 : N_1$ do
 8:             $v^{(k)}(((i-1)\cdot N_2+1) : i\cdot N_2) = \hat{v}^{(k)}(((i-1)\cdot N_2+1) : i\cdot N_2)/\|\hat{v}^{(k)}(((i-1)\cdot N_2+1) : i\cdot N_2)\|_1$
 9:         end
10:     end
11:     $k = k+1$;
12: until convergence;
13: $\boldsymbol{u}^{(k)} = \boldsymbol{v}^{2(k)}$
    Note: $\boldsymbol{u}^{(k)} = \boldsymbol{v}^{2(k)}$, and $v^{(k)}(((i-1)\cdot N_2+1) : i\cdot N_2)$ denotes the slice of $v^{(k)}$ with indices from $(i-1)\cdot N_2+1$ to $i\cdot N_2$.
```

4.2 Supersymmetric Higher-order Power Iteration

The higher-order tensor problem in Eq. (1) may be solved by tensor decomposition [32]; tensor decomposition originated in [33]. We utilize the rank-one higher-order power method [34] to approximately solve Eq. (1); as noted, an exact computation is infeasible. Eq. (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 $\boldsymbol{x}\in\{0,1\}^N$, and $\star$ is the *Tucker product* [23], which is an N-fold succession of products. To get an approximate solution, we relax the constraints: the binary assignment vector $\boldsymbol{x}\in\{0,1\}^N$ is replaced by an assignment vector $\boldsymbol{u}$ with real elements in $[0,1]$. These values are naturally computed from the potential functions in Section 4.4. This changes the optimization problem to one of computing the rank-one approximation of the affinity tensor $\mathcal{T}_N$ [23], i.e. finding a scalar λ and a unit norm vector $\boldsymbol{u}\in\mathbb{R}^N$, 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 Frobenius norm squared function $f(\hat{\mathcal{T}_N}) = \|\mathcal{T}_r - \hat{\mathcal{T}_N}\|_F^2$. 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 for finding rank-one tensor approximations; a version for supersymmetric tensors (S-HOPM) is given in [23]. The S-HOPM algorithm converges under the assumption of convexity (or concavity) for the functional induced by the tensor [23], which is sufficiently robust for practical applications. S-HOPM is performed in two iterative steps: higher-order power iteration of $\boldsymbol{u}$, followed by normalization of $\boldsymbol{u}$ under the Frobenius norm. A recent effective improvement [21], [29] uses the ℓ^1 norm to replace the traditional ℓ^2 norm.

We use the ℓ^1 norm, and further revise S-HOPM as follows. To perform higher-order power iteration with $\boldsymbol{u}$, we must compute $\hat{\boldsymbol{u}}^{(k)} = \mathcal{I} \overset{\mathcal{T}_N}{\star} (\boldsymbol{u}^{(k-1)})^{\overset{\mathcal{T}_N}{\star}(N-1)}$, where $\overset{\mathcal{T}_N}{\star}$ is a so-called $\mathcal{T}_N$-product, and $\mathcal{I}$ is a unit tensor [23]. In the specific case that $\hat{\boldsymbol{u}}^{(k)}$ belongs to an N^{th}-order supersymmetric affinity tensor, it can be shown [23] that $\hat{\boldsymbol{u}}^{(k)} = \mathcal{I} \overset{\mathcal{T}_N}{\star} (\boldsymbol{u}^{(k-1)})^{\overset{\mathcal{T}_N}{\star}(N-1)}$ implies that $\forall m \in (i_1, ..., i_N)$, $v_m^{(k)} =$

$$\sum_{i_1,...,i_N} T_N(i_1,...,i_N) 2 v_m^{(k-1)} v_{i_1}^{2(k-1)} ... v_{m-1}^{2(k-1)} v_{m+1}^{2(k-1)} ... v_{i_N}^{2(k-1)}, \tag{4}$$

where $\boldsymbol{u}^{(k)} = \boldsymbol{v}^{2(k)}$. For the supersymmetric affinity tensor, putting Definition 2 into the former expression leads to (see Appendix for derivation) that $v_m^{(k)} =$

$$(N–1)! \phi_N(i_1,...,i_N) 2 v_m^{(k-1)} v_{i_1}^{2(k-1)} ... v_{m-1}^{2(k-1)} v_{m+1}^{2(k-1)} ... v_{i_N}^{2(k-1)}, \tag{5}$$

where ϕ_N is the potential function explained in Section 4.4. Eq. (5) is more compact than earlier expressions in the literature, as it handles all symmetrically related potential elements as a single item using multiplication by $(N-1)!$.

Many initialization schemes have been proposed for the S-HOPM method [23]. We simply use positive random values between 0 and 1 to initialize $\boldsymbol{u}_0$, which ensures convergence; proofs are detailed in [23], [35].

Our supersymmetric higher-order power iteration solution of Eq. (1) is performed by the SuperMatching algorithm—see Algorithm 1. Its efficiency relies on two principles. Firstly, we take advantage of the supersymmetry to deduce $\boldsymbol{u}$ as in Eq. (5), using just a single canonical element for computation (see Step 5). Secondly, power iteration just considers potential elements, and excludes all zero elements from the iteration process. The complexity of the whole iteration process depends only on the number $|\theta_N|$ of non-zero affinities. Consequently, this method reduces also memory costs while keeping accuracy.

Note that, although [21], [29] use a supersymmetric affinity tensor, their approaches do not make full use of supersymmetry when creating the supersymmetric affinity tensor, nor do they take advantage of supersymmetry to accelerate the power iteration process. By doing so, we overcome limitations due to unbalanced and redundant tensor elements in [21], [29], as our experiments show later.

4.3 Sampling Strategy

Algorithm 1 depends on the potential elements. We next discuss the issue of how to sample the feature tuples to build potential elements, which determines the size $|\theta_N|$ and influences matching accuracy (as well as speed).

Given two feature sets P_1 and P_2, a potential element may be obtained by using a feature tuple sampled from each feature set separately. For N^{th}-order matching, a straightforward way to construct the potential elements is: first find all feature tuples for P_1 and P_2, as F_1 and F_2; then $\forall (f_{i_1}^1, \cdots, f_{i_N}^1) \in F_1$, calculate the potentials for $(f_{i_1}^1, \cdots, f_{i_N}^1)$ with all feature tuples in F_2. This is far too time-consuming, so sampling is used instead. We suggest random sampling for general feature matching problems, but this does not preclude more directed sampling if prior knowledge of the matching indicates a better approach.

Our sampling technique repeatedly randomly samples t_1 feature tuples for each feature point from P_1, and fully samples P_2. For P_1, we take each feature in turn as a required element, and then randomly choose t_1 feature tuples containing this required element. Thus, the number of feature tuples in F_1 is $N_1 t_1$, and N_2^N in F_2. Then, for each feature tuple in F_1, we find the k most similar feature tuples 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 $\theta_N = \{\phi_i^k\}_{i=1}^{N_1 t_1}$, of size $|\theta_N| = N_1 t_1 k$. For P_1, the sampling cost is $O(N_1 t_1 k \log N_2)$, where the $\log N_2$ factor comes from use of a k-D tree to search for the k most similar feature tuples in F_2. The parameters t_1 and k must be chosen according to the size of the feature sets. In practice, for two feature sets each with hundreds points, we may take $t_1 \approx 100$ and $k \approx 300$ for third-order matching. Our experiments demonstrate that this sampling approach works well.

An important aspect of our sampling approach is to use the supersymmetry of the affinity tensor. Potential elements whose indices are permutations of each other have the same value, so should not be repeatedly sampled. Thus, we use a sampling constraint that the sets of feature tuples F_1 obtained from the sampling process should have no repetition, in the sense that

$$\begin{aligned} &\forall (f_{i_1}^1, f_{i_2}^1, \cdots, f_{i_N}^1), (f_{j_1}^1, f_{j_2}^1, \cdots, f_{j_N}^1) \in F_1, \\ &\quad (f_{i_1}^1, f_{i_2}^1, \cdots, f_{i_N}^1) \neq \Omega(f_{j_1}^1, f_{j_2}^1, \cdots, f_{j_N}^1) \end{aligned} \tag{6}$$

where Ω is an arbitrary permutation.

Earlier work [21], [22], [28] adopted random sampling, but failed to impose any constraint on the sampling process to take into account supersymmetry, leading to the possibility that feature tuples may be sampled multiple times. For example, for third-order matching, it is possible that a feature tuple $(f^1_{i_1}, f^1_{i_2}, f^1_{i_3})$ may be sampled from P_1 and $(f^2_{i_1}, f^2_{i_2}, f^2_{i_3})$ from P_2, and also a feature tuple $(f^1_{i_1}, f^1_{i_3}, f^1_{i_2})$ from P_1 and $(f^2_{i_1}, f^2_{i_3}, f^2_{i_2})$ from P_2. That would create two tensor elements $\phi_3(i_1, i_2, i_3)$ with index (i_1, i_2, i_3) and $\phi_3(i_1, i_3, i_2)$ with index (i_1, i_3, i_2), which are the same. However, we just need one tensor element to express the affinity on the assignment group (i_1, i_2, i_3) for any permutation of indices. Such extra sampling is not only inefficient, but may also reduce the accuracy of the power iteration: one set of symmetrically related elements may be represented by a different number of samples than another set of symmetrically related elements, which imbalances the power iteration process, and can lead to inaccurate results. Our sampling method reduces the sampling cost, while also improving the accuracy of power iteration.

4.4 Higher-order Potentials

Different higher-order potentials are appropriate for different applications. Here we briefly give two simple examples of general higher-order potentials for 2D and 3D matching respectively; we use them later to evaluate our algorithm. The potentials are based on a Gaussian function which guarantees the tensor elements are non-negative and invariant under any permutation of the input assignments.

In 2D, we use a well-known [21]–[30] third-order geometric-similarity invariant potential ϕ_3 for linking point feature triples. Triangles formed by three points are *similar* under scaling, rotation and translation—interior angles are invariant. Thus ϕ_3 can be defined in terms of differences of corresponding interior angles:

$$\begin{aligned} \phi_3(i_1, i_2, i_3) &= \phi_3(\{p_1, q_1\}, \{p_2, q_2\}, \{p_3, q_3\}) \\ &= \exp(-1/\varepsilon^2 \textstyle\sum_{(l,l')} \|\alpha_l - \alpha_{l'}\|^2) \quad (7) \end{aligned}$$

where $\varepsilon > 0$ is the kernel bandwidth, which is simply set to the average of the ℓ^1 norm of all differences, and $\{\alpha_l\}^3_{l=1}$ and $\{\alpha^{'}_l\}^3_{l'=1'}$ are the angles formed by feature triples (p_1, p_2, p_3) and (q_1, q_2, q_3): see Figure 2. Each point corresponds to one interior angle. This potential may be extended to higher order by using the internal angles of polygons with more than 3 sides.

For 3D matching problems, we may replace the internal angles by edge lengths, which for meshes are based on geodesic distance across the mesh in which the points are embedded. This corresponds to assuming an isometry transform relating the meshes. Geodesic distance may be computed by Dijkstra's algorithm [36], based on Euclidean distances between neighboring mesh vertices. See Figure 2.

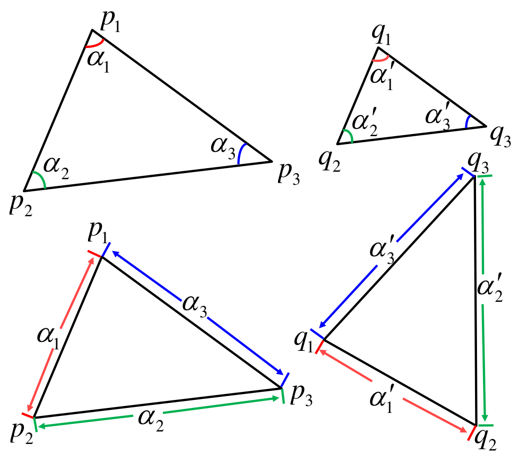


Fig. 2. Third-order potential. The geometric constraints are: internal angle invariance in 2D (above), and edge length invariance in 3D (below).

5 Experiments

We have used synthetic data and real captured data to evaluate the SuperMatching algorithm on a 2.3GHz Core2Duo PC with 2GB memory.

To demonstrate the independence of the algorithm from choice of feature descriptors, several descriptors were used. In some cases, we simply uniformly sampled feature points on the objects, and used a trivial feature descriptor of 1 for all points, meaning affinities are based simply on distances between feature points—this allows us to show our method is robust in the presence of ambiguous feature descriptors. In other cases, we still used uniformly distributed feature points, together with SIFT descriptors, which shows that feature points do not have to be carefully chosen. *Note that we only used the feature descriptors to select points as feature points and did not employ them to build the matching.* Both uniform and SIFT features are widely used in real applications. Choosing descriptors in this way lets us show that the SuperMatching is independent of the descriptors to a fair degree. The feature descriptors could be replaced by any others—many have been proposed. We used third-order matching in our experiments; it would be simple (but more costly) to use higher order.

5.1 3D Rigid Shapes Scans

Firstly, we used SuperMatching to build pairwise matchings between 3D rigid shape scans based on uniformly sampled feature points. Rigid transforms can be computed from each triple of compatible matching points. The transform which brings the most data points within a threshold distance of a point in the model is chosen as the optimal alignment transform [9]. As discussed in [38], such a voting

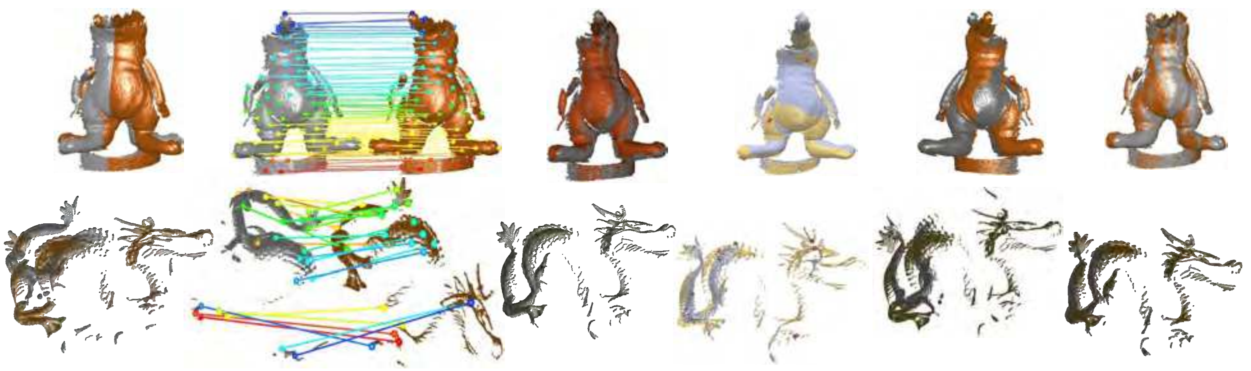


Fig. 3. Pairwise alignment of Coati and Dragon scans. From left to right: before alignment, our matching result, our alignment result, and alignment results from 4PCS [31], ICP [37], [21], and [29].

scheme is guaranteed to find the optimal alignment between pairwise scans and is robust to the initial pose of the input scans.

To test the robustness of SuperMatching, the noisy shapes used to test the recent 4PCS rigid registration algorithm [31] were considered. The results demonstrate that SuperMatching can produce good registration for all of these shapes. For example, Figure 3 shows the initial state of the input, and registration results based on SuperMatching and 4PCS [31]. Even for the Coati and Dragon examples, it is noticeable that the final alignment details from the SuperMatching are somewhat better than for 4PCS. Other results produced by ICP [37], [21], and [29] are shown in Figure 3. It is noticeable that ICP [37] failed, while the results from Supermatching, [21], and [29] are almost the same. The times taken by both SuperMatching and the 4PCS algorithm were very similar (about 10 seconds in each case), but [21] and [29] took over 20 seconds. Our time performance benefits from the compact representation in Eq. (5) and the sampling strategy based on supersymmetry.

Figure 4 shows some registration results for the Rooster model from [39]. The left column shows the original state, the two left middle columns are our matching and alignment results, and the right columns shows the corresponding results produced by 4PCS [31], ICP [37], [21], and [29]. In this case, 4PCS [31] and ICP [37] have clearly failed. We believe the main reason for failure of 4PCS is that it does not take into account the correlation between each set of 4 congruent points, and just computes optimal alignment of sets. On the other hand, SuperMatching uses a global optimization scheme taking into account all higher-order feature tuples at once to compute the matching, as formulated by Eq. (1). The transformation matrix computed by SuperMatching is compared with the ground truth provided by [39] in Table 1, demonstrating that our computed matrices for matching I-II and I-III are close to the ground truth. (The matrices are those needed to transform and align from II to I and III to I).

TABLE 1
Transformation matrix comparison.

	Computed matrix				Ground truth			
I	0.730	-0.005	-0.682	-127.6	0.707	-0.004	-0.707	-132.0
—	-0.006	1.000	-0.016	-3.016	-0.012	1.000	-0.018	-3.300
II	0.693	0.021	0.700	-50.05	0.707	0.021	0.707	-54.10
	0	0	0	1	0	0	0	1
I	0.005	-0.012	-0.995	-186.8	0.001	-0.022	-1.000	-187.7
—	-0.014	1.003	-0.022	-4.769	-0.032	0.999	-0.022	-4.100
III	1.017	0.036	0.005	-176.6	1.000	0.032	0.000	-186.0
	0	0	0	1	0	0	0	1

TABLE 2
The closest point-to-point standard RMS errors.

Shape pairs	SuperMatching	4PCS [31]	ICP [37]	[21], [29]
Coati	0.410	0.589	1.231	0.457
Dragon	0.438	1.013	2.576	0.500
Rooster I-II	0.276	0.376	0.386	0.280
Rooster II-III	0.310	0.333	0.337	0.312

Estimation error was measured using the closest point-to-point standard RMS error between the final pairs. For these four pairs (Coati, Dragon, Rooster I-II, Rooster II-III), Table 2 lists the RMS errors. SuperMatching outperforms 4PCS [31], ICP [37], [21], and [29] when matching partial, noisy data.

Next, we used SuperMatching to build a complete model from a set of scans from different viewpoints. For these multiple scans, third-order matching was first performed between each pair of consecutive scans. After doing so, the alignment was refined using the iterative closest point (ICP) algorithm [37]. Figure 5 illustrates the approach and shows the result.

5.2 3D Depth Scans with Color Information

We next provide a further real-world, noisy, example of the use of SuperMatching. In this case, data with surface color information was captured using a Kinect camera [40]. Uniform samples and SIFT feature

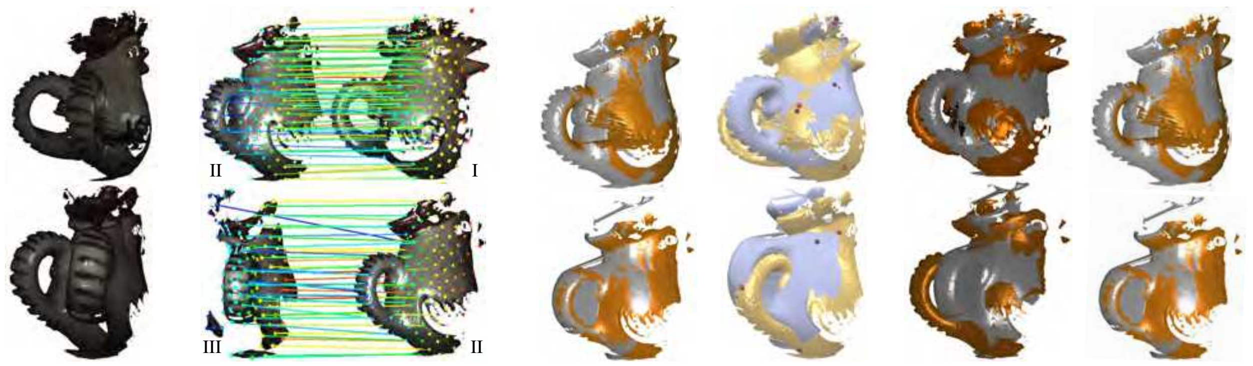


Fig. 4. Pairwise alignment of Rooster I-II and II-III scans. From left to right: before alignment, our matching result, our alignment result, and alignment results from 4PCS [31], ICP [37], [21], and [29].

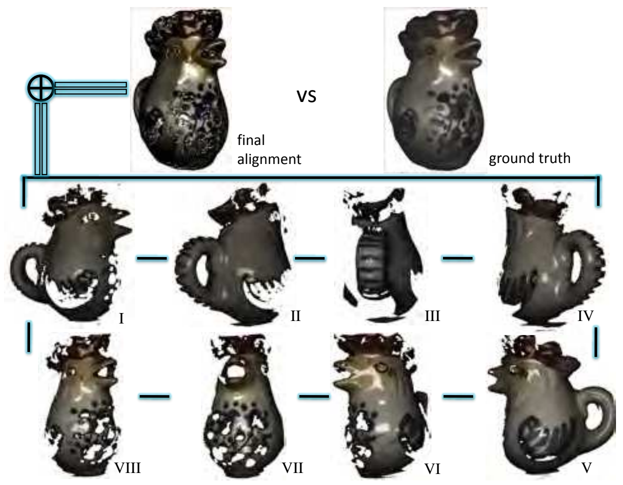


Fig. 5. Alignment of several Rooster scans from different viewpoints. Above: our final registered Rooster and the ground truth [39]. Below: 8 partial scans, the dark lines indicating the pairwise matching process.

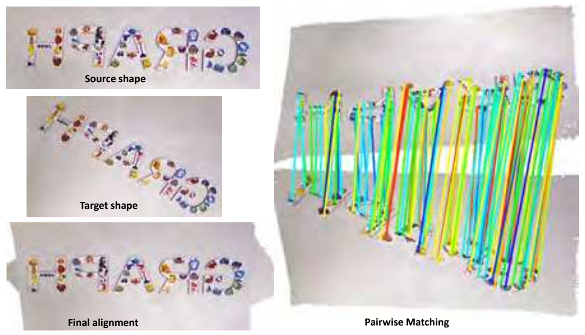


Fig. 6. 3D real depth scans with color information, captured using Kinect. Matching points are connected by colored lines.

vectors were used as a basis for SuperMatching. This resulted in robust matches, as illustrated in Figure 6. (Source and target shapes are reversed, as the Kinect device mirrors the input scene). There are no matching points on the flat parts shown in white, as these lack SIFT features and are filtered out before matching.

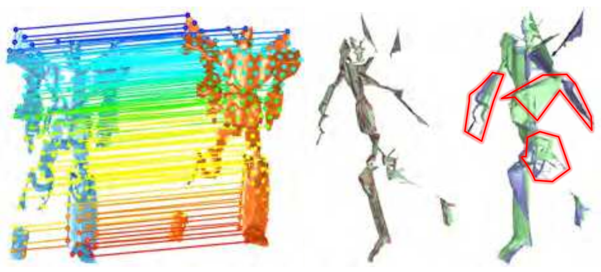


Fig. 7. Pairwise matching of an articulated Robot between two frames. Left: our matching. Right: registered results produced by our approach and [42], side view; red polygons indicate regions of large distortion.

5.3 3D Articulated Shape Synthetic Data

A further application is registration of approximately articulated shapes. Such problems are common in dynamic range scanning such as human motion capture. Given a sequence of range scans of a moving articulated subject, our method automatically registers all data to produce a complete 3D shape. Unlike many other methods, we do not need manual segmentation, markers, or a prior template. As shown by Figure 7, SuperMatching provides robust, accurate matching, even although the partial scans have holes and different poses.

Again uniformly sampled points were used. Registration of scans was performed by computing piecewise rigid transformations between matches. Each triple provides a single rigid transformation; we then used the mean-shift algorithm to cluster the transformations [41]. These transformations may be propagated from feature points to the entire set of points in each scan using nearest neighbor interpolation. Figure 7 shows a registration example for an articulated model, and in comparison a result produced by the method in [42]. Our matching and registration results are more accurate and plausible.

5.4 Isometric Deformable Surfaces

We further matched SIFT points on images of deforming surfaces[1] showing a cloth and a cushion. The surface of the cloth underwent relatively smooth deformation, while the surface of the cushion included sharp folds. This data comes with ground truth, which allows quantitative verification of the accuracy of the matches found. From each surface set we randomly chose two frames before and after a large deformation. We randomly chose 100 corresponding points on each surface, using the provided ground truth.

We used the above input data as a basis for comparison with a spectral algorithm [26] (a quadratic assignment algorithm), the third-order tensor algorithm in [21], [29], and a hypergraph matching algorithm [28], using the authors' code in each case. All methods were executed in Matlab using the PC configuration previously given. To enable direct and fair comparison, [21], [29], [28] and SuperMatching used the same potential and the same tensor size.

In these tests, SuperMatching considered 3×10^6 feature tuples, while the methods in [21], [29] considered 10×10^6 tuples and the method of [28] used 4×10^6. The difference in tuple numbers mainly comes from the sampling strategy, but we have the lowest sampling cost. The extra sampling of [21], [29] relative to ours is because they have no supersymmetric constraint on the sampling process, so about $2/3$ of their samples are redundant, and extra samples are needed to make the comparison with our approach fair. For similar reasons, we also used more tuples to evaluate [28]. The average running time to match two feature sets each with 100 features was around 8s for SuperMatching, 13s for [21], 6.5s for [28], and 5s for [26]. SuperMatching takes less time than the third-order tensor algorithm [21], [29] both because it uses fewer feature tuples and because of the more efficient supersymmetric higher-order power iteration solution.

Matching accuracy was assessed by the number of correctly matched points (known from the ground truth) divided by the total number of points that could be matched. The results are summarised in Table 3 and illustrated in Figure 8. Table 3 demonstrates that SuperMatching achieves a higher matching accuracy than previous algorithms. The worst matching result is produced by the spectral quadratic assignment algorithm [26], due to the lower discriminatory power of pairwise geometric constraints. Higher-order algorithms perform better due to the more complex geometric constraints. Nevertheless, SuperMatching significantly outperforms other third-order algorithms [21], [29] and the hypergraph matching algorithm [28].

1. From http://cvlab.epfl.ch/data/dsr/

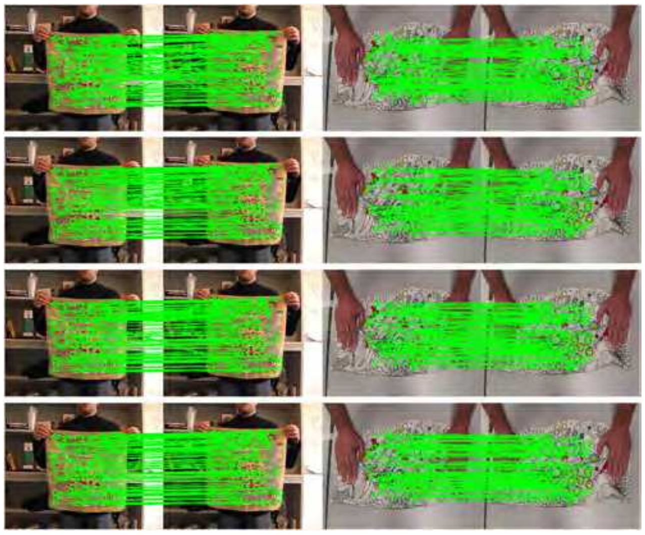


Fig. 8. Matching results. Left: cloth set, matching between frame 80 and 90, right: cushion set, matching between 144 and 156. Top to bottom, spectral method [26], hypergraph matching method [28], a third-order tensor method [21], [29], and SuperMatching.

TABLE 3
Accuracy of deformable surface matching.

Dataset	cloth				cushion				
Matching frames	F80-F90	F90-F95	F95-F100	F100-F105	F144-F156	F156-F165	F165-F172	F172-F188	Time
Ours	83%	85%	84%	81%	66%	60%	69%	56%	8s
[28]	73%	79%	70%	72%	44%	39%	54%	43%	6.5s
[21], [29]	67%	77%	73%	65%	39%	31%	47%	42%	13s
[26]	27%	29%	22%	27%	14%	5%	28%	7%	5s

5.5 Image matching under affine transformation

Previous examples used third-order potentials ($N = 3$), and as we show here, higher order potentials ($N > 3$) offer more accurate matching at the cost of extra running time. Matching images related by an affine transformation is an important task, and here a fourth-order potential function specific to this particular matching application is appropriate.

Our new fourth-order potential ϕ_4 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 (see Fig. 9 top-left) as:

$$\begin{aligned} \phi_4(i_1, i_2, i_3, i_4) &= \phi_4(\{p_1, q_1\}, \{p_2, q_2\}, \{p_3, q_3\}, \{p_4, q_4\}) \\ &= \exp(-1/\varepsilon^2 \sum\nolimits_{(l,l')} \|\alpha_l - \alpha_{l'}\|^2) \qquad (8) \end{aligned}$$

where $\{\alpha_l\}_{l=1}^4$ and $\{\alpha_l^{'}\}_{l'=1}^4$ 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 $\alpha_1 = S_{\triangle p_1 p_2 p_3}/S_{\square p_1 p_2 p_3 p_4}$, $\alpha_2 = S_{\triangle p_2 p_3 p_4}/S_{\square p_1 p_2 p_3 p_4}$, $\alpha_3 = S_{\triangle p_1 p_3 p_4}/S_{\square p_1 p_2 p_3 p_4}$, $\alpha_4 = S_{\triangle p_1 p_2 p_4}/S_{\square p_1 p_2 p_3 p_4}$, and similarly for the other quadrilaterals.

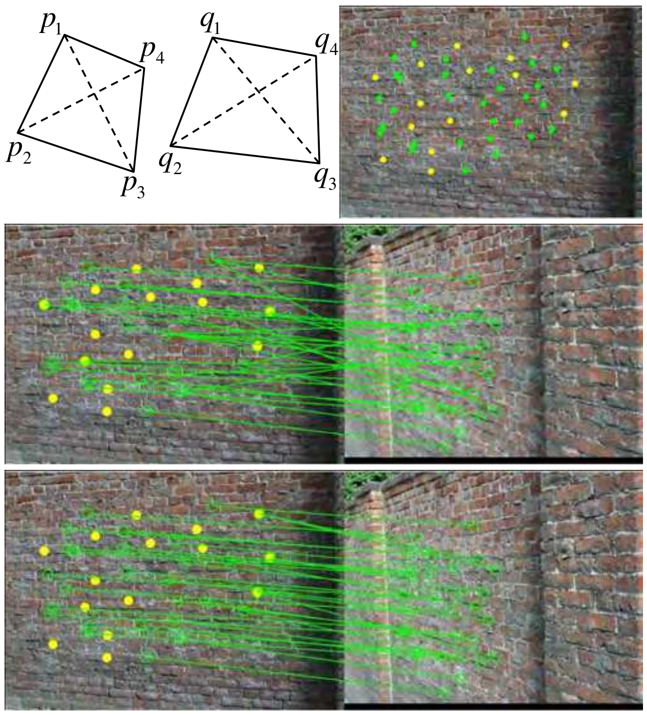


Fig. 9. Higher-order image matching under affine transformation. Top-left: construction of fourth-order potential; Top-right: image 1 in wall set; features detected by MSER shown in green, yellow points are outliers. Middle, bottom: results using third and fourth order matching.

We now use third- and fourth-order potentials to evaluate our algorithm; for third-order we use the potential function in Eq. (7) based on internal angle invariance in 2D. We used the `wall`[2] image set to evaluate the matching accuracy of our method under affine transformations. For the test images, we used 30 feature points detected by MSER [43] in the central area of image 1 as feature set P_1 (the green points in Figure 9 top-right). In the test, the images were taken from quite different viewpoints, leading to quite different texture appearances, making it difficult to match features just using an MSER or SIFT detector. The results demonstrate that both third (Figure 9 middle) and fourth (Figure 9 bottom) methods produce a high matching accuracy as they take structural information into account. However, the fourth-order method is much more stable than third-order, at the cost of higher computation time (about 6s vs 4s).

6 Conclusions

This paper has presented the SuperMatching algorithm, which tackles the classic computer graphics and computer vision problem of feature matching, independently of feature description. It is an efficient higher-order matching algorithm which uses a compact form of the higher-order supersymmetric affinity tensor to express relatedness of features. Matching is performed using an efficient power iteration method, which takes advantage of supersymmetry and avoids computing with zero elements. We also give an efficient sampling strategy for choosing feature tuples to create the affinity tensor. Experiments on both synthetic and real 2D and 3D data sets show that SuperMatching has greater accuracy than competing methods, whilst having competitive performance.

In future, we wish to further improve the performance of the SuperMatching algorithm. Random sampling could be executed in parallel. We also intend to apply it to more challenging imperfect deformable 3D data matching based on real captured data. SuperMatching is applicable to many fields, as matching is a foundation for many computer graphics and computer vision applications.

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

Appendix
Proof of Eq.5 in the Third-order Case

$\forall (i_1, i_2, i_3) \in \theta_3,$

$$\begin{aligned}
v_{i_1}^{(k)} =& \mathcal{T}_3(i_1,i_2,i_3)2v_{i_1}^{(k-1)}v_{i_2}^{2(k-1)}v_{i_3}^{2(k-1)} \\
&+ \mathcal{T}_3(i_1,i_3,i_2)2v_{i_1}^{(k-1)}v_{i_2}^{2(k-1)}v_{i_3}^{2(k-1)} \\
=& \phi_3(i_1,i_2,i_3)2v_{i_1}^{(k-1)}v_{i_2}^{2(k-1)}v_{i_3}^{2(k-1)} \\
&+ \phi_3(i_1,i_3,i_2)2v_{i_1}^{(k-1)}v_{i_2}^{2(k-1)}v_{i_3}^{2(k-1))} \\
=& 2\phi_3(i_1,i_2,i_3)2v_{i_1}^{(k-1)}v_{i_2}^{2(k-1)}v_{i_3}^{2(k-1)}.
\end{aligned}$$

$$\begin{aligned}
v_{i_2}^{(k)} =& \mathcal{T}_3(i_2,i_1,i_3)2v_{i_2}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_3}^{2(k-1)} \\
&+ \mathcal{T}_3(i_2,i_3,i_1)2v_{i_2}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_3}^{2(k-1)} \\
=& \phi_3(i_2,i_1,i_3)2v_{i_2}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_3}^{2(k-1)} \\
&+ \phi_3(i_2,i_3,i_1)2v_{i_2}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_3}^{2(k-1)} \\
=& 2\phi_3(i_2,i_1,i_3)2v_{i_2}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_3}^{2(k-1)}.
\end{aligned}$$

$$\begin{aligned}
v_{i_3}^{(k)} =& \mathcal{T}_3(i_3,i_1,i_2)2v_{i_3}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_2}^{2(k-1)} \\
&+ \mathcal{T}_3(i_3,i_2,i_1)2v_{i_3}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_2}^{2(k-1)} \\
=& \phi_3(i_3,i_1,i_2)2v_{i_3}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_2}^{2(k-1)} \\
&+ \phi_3(i_3,i_2,i_1)2v_{i_3}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_2}^{2(k-1)} \\
=& 2\phi_3(i_3,i_1,i_2)2v_{i_3}^{(k-1)}v_{i_1}^{2(k-1)}v_{i_2}^{2(k-1)}.
\end{aligned}$$

Acknowledgments

We are grateful to Will Chang, Timothée Cour, Olivier Duchenne, Misha Kazhdan, Niloy J. Mitra, and Ron Zass, for sharing source code, executable programs, and test data.

This work was supported by the Natural Science Foundation of China (No. 61103084, 61272334).

REFERENCES

[1] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, 1992.

[2] W. Chang and M. Zwicker, "Global registration of dynamic range scans for articulated model reconstruction," *ACM Transactions on Graphics*, vol. 30, pp. 26:1–26:15, 2011.

[3] 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*, vol. 30, pp. 1:1–1:20, 2011.

[4] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondence," in *IEEE CVPR*, pp. 26–33, 2005.

[5] T. Windheuser, U. Schlickwei, F. R. Schimdt, and D. Cremers, "Large-scale integer linear programming for orientation preserving 3d shape matching," *Computer Graphics Forum (Proc. SGP)*, vol. 30, no. 5, pp. 1471–1480, 2011.

[6] B. J. Brown and S. Rusinkiewicz, "Global non-rigid alignment of 3-d scans," *ACM Transactions on Graphics*, vol. 26, 2007.

[7] Y. Pekelny and C. Gotsman, "Articulated object reconstruction and markerless motion capture from depth video," *Computer Graphics Forum (Proc. EuroGraphics)*, vol. 27, no. 2, pp. 399–408, 2008.

[8] M. Wand, B. Adams, M. Ovsjanikov, A. Berner, M. Bokeloh, P. Jenke, L. Guibas, H.-P. Seidel, and A. Schilling, "Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data," *ACM Transactions on Graphics*, vol. 28, pp. 15:1–15:15, 2009.

[9] D. P. Huttenlocher and S. Ullman, "Recognizing solid objects by alignment with an image," *International Journal of Computer Vision*, vol. 5, pp. 195–212, November 1990.

[10] V. G. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," in *SIGGRAPH*, pp. 79:1–79:12, 2011.

[11] 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*, vol. 21, no. 5, pp. 433–449, 1999.

[12] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *International Conference of Computer Vision*, 2011.

[13] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1681–1707, 2011.

[14] A. Tevs, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel, "Isometric registration of ambiguous and partial data," in *IEEE CVPR*, pp. 1185–1192, 2009.

[15] A. Tevs, A. Berner, M. Wand, I. Ihrke, and H.-P. Seidel, "Intrinsic shape matching by planned landmark sampling," *Computer Graphics Forum*, vol. 30, no. 2, pp. 543–552, 2011.

[16] M. Ovsjanikov, Q. Mrigot, F. Mmoli, and L. Guibas, "One point isometric matching with the heat kernel," *Computer Graphics Forum (Proc. SGP)*, vol. 29, no. 5, pp. 1555–1564, 2010.

[17] Y. Lipman and T. Funkhouser, "Möbius voting for surface correspondence," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, pp. 72:1–72:12, 2009.

[18] 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*, vol. 18, no. 3, pp. 265–298, 2004.

[19] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *International Conference of Computer Vision*, pp. 1482–1489, 2005.

[20] Y. Zeng, C. Wang, Y. Wang, X. Gu, D. Samaras, and N. Paragios, "Dense non-rigid surface registration using high-order graph matching," in *IEEE CVPR*, pp. 382–389, 2010.

[21] O. Duchenne, F. Bach, I. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," in *IEEE CVPR*, pp. 1980–1987, 2009.

[22] A. Wang, S. Li, and L. Zeng, "Multiple order graph matching," in *Asian Conference on Computer Vision*, pp. 471–482, 2010.

[23] E. Kofidis and P. A. Regalia, "On the best rank-1 approximation of higher-order supersymmetric tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 863–884, 2002.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[25] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Symposium on Geometry Processing*, pp. 1383–1392, 2009.

[26] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in *NIPS*, pp. 313–320, 2006.

[27] H. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans," *Computer Graphics Forum (Proc. SGP)*, vol. 27, no. 5, pp. 1421–1430, 2008.

[28] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *IEEE CVPR*, pp. 1–8, 2008.

[29] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 2383–2395, 2011.

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

[31] D. Aiger, N. J. Mitra, and D. Cohen-Or, "4-points congruent sets for robust pairwise surface registration," *ACM Transactions on Graphics*, vol. 27, no. 3, 2008.

[32] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[33] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, pp. 064–089, 1927.

[34] L. D. Lathauwer, P. Comon, B. D. Moor, and J. Vandewalle, "Higher-order power method," in *Proceedings of NOLTA*, pp. 2709–2712, 1995.

[35] P. A. Regalia and E. Kofidis, "The higher-order power method revisited: convergence proofs and effective initialization," in *Proceedings of the Acoustics Speech and Signal Processing*, pp. 2709–2712, IEEE Computer Society, 2000.

[36] G. Peyré, M. Péchaud, R. Keriven, and L. D. Cohen, "Geodesic methods in computer vision and graphics," *Foundations and Trends in Computer Graphics and Vision*, vol. 5, pp. 197–397, 2010.

[37] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3DIM*, pp. 145–152, IEEE Computer Society, 2001.

[38] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust global registration," in *Symposium on Geometry processing*, 2005.

[39] M. Chuang, L. Luo, B. J. Brown, S. Rusinkiewicz, and M. M. Kazhdan, "Estimating the laplace-beltrami operator by restricting 3d functions," *Computer Graphics Forum (Proc. SGP)*, vol. 28, no. 5, pp. 1475–1484, 2009.

[40] Kinect, "Kinect homepage," 2012. http://www.xbox.com/en-US/kinect.

[41] N. J. Mitra, L. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3d geometry," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 25, no. 3, pp. 560–568, 2006.

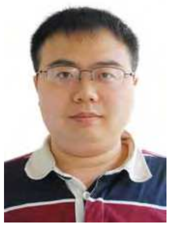
[42] W. Chang and M. Zwicker, "Range scan registration using reduced deformable models," *Computer Graphics Forum (Proc. Eurographics)*, vol. 28, no. 2, pp. 447–456, 2009.

[43] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

Zhi-Quan Cheng received a BSc, MSc, and PhD degree from Computer School at National University of Defense Technology in 2000, 2002 and 2008, respectively. He is lecturer at the PDL Laboratory, Computer School, National University of Defense Technology (NUDT), and the leader of visual computing team at NUDT. His research interests include computer graphics, and digital geometry processing.

Yin Chen received a BSc and MSc degree from Computer School at National University of Defense Technology in 2008 and 2010, respectively. He is a Ph.D student, Computer School, National University of Defense Technology (NUDT). His research interests include computer graphics, and digital geometry processing.

Ralph R. Martin Ralph R. Martin received the PhD from Cambridge University in 1983, with a dissertation on Principal Patches, and since then, has worked his way up from a lecturer to a professor at Cardiff University. He has been working in the field of CADCAM since 1979. He has published more than 170 papers and 10 books covering such topics as solid modelling, surface modelling, intelligent sketch input, vision based geometric inspection, geometric reasoning and reverse engineering. He is a fellow of the Institute of Mathematics and Its Applications, and a member of the British Computer Society. He is on the editorial boards of Computer Aided Design, Computer Aided Geometric Design, the International Journal of Shape Modelling, the International Journal of CADCAM, and Computer- Aided Design and Applications. He has also been active in the organisation of many conferences.



Yu-Kun Lai received his bachelors degree and PhD degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a lecturer of visual computing in the School of Computer Science, Cardiff University, Wales, UK. His research interests include computer graphics, geometry processing, computer-aided geometric design and computer vision.

Aiping Wang received a BSc, MSc, and PhD degree from Computer School at National University of Defense Technology in 2004, 2006 and 2011, respectively. He is lecturer Computer School, National University of Defense Technology (NUDT). His research interests include computer graphics and vision.