SuperMatching: Feature Matching using Supersymmetric Geometric Constraints

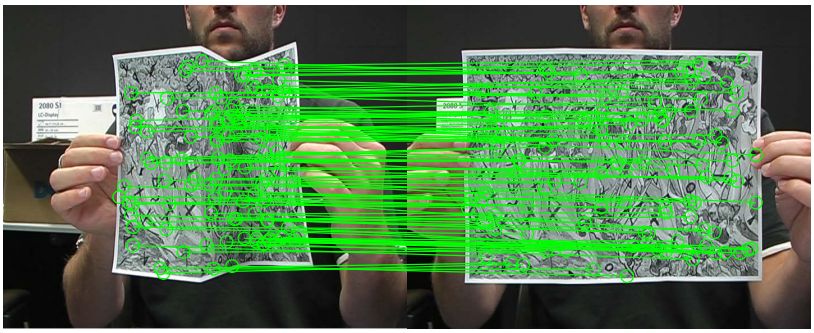

Figure 1: *Correspondences between 3D datasets determined by SuperMatching, using combined SIFT features (for color) and slippage features (for geometry). Left: rigid pairwise matching of views of a colored jug. Right: matching of a unfolded piece of paper with a large deformation.*

Abstract

Feature matching is a challenging problem lying at the heart of numerous computer graphics and computer vision applications. We present here the *SuperMatching* algorithm for finding correspondences between two sets of features. It does so by considering triangles 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 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 tensor compactly. Matching is performed by an efficient higher-order power iteration approache which takes advantage of the compact representation of the supersymmetric affinity tensor. Experiments on both synthetic and real captured data show that SuperMatching provides accurate feature matching for a wide range of 2D and 3D features, giving more reliable results than other state-of-the-art approaches, with competitive computational cost.

Keywords: Feature matching; Geometric constraints; Supersymmetric tensor

1 Introduction

Building correspondences between two sets of features belonging to a pair of 2D 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 [Besl and McKay 1992; Gelfand et al. 2005; Aiger et al. 2008; Li et al. 2008; Chang and Zwicker 2009; Zeng et al. 2010; van Kaick et al. 2011; Chang and Zwicker 2011], shape retrieval from databases [Bronstein et al. 2011], shape matching [Berg et al. 2005; Brown and Rusinkiewicz 2007; Torresani et al. 2008; Tevs et al. 2009; Ovsjanikov et al. 2010; Tevs et al. 2011; Sahillioglu and Yemez 2011; Windheuser et al. 2011], shape reconstruction [Brown and Rusinkiewicz 2007; Pekelny and Gotsman 2008; Wand et al. 2009; Chang and Zwicker 2011], and automatic shape understanding [Lipman and Funkhouser 2009; Sun et al. 2010; Kim et al. 2011] .

In principle and practice, determining correspondences is typically done in three steps [Johnson and Hebert 1999; Lowe 2004; Sun et al. 2009; Bokeloh et al. 2008; Toler-Franklin et al. 2010; Leutenegger et al. 2011]: (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 as their importance is easily perceivable. 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 [van Kaick et al. 2011]: 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 [Tevs et al. 2009; Tevs et al. 2011] minimize the effects of outliers, generalized multidimensional scaling [Bronstein et al. 2011] and heat kernel maps [Ovsjanikov et al. 2010] consider the manifold in which the points are embedded. Möbius transformations [Lipman and Funkhouser 2009; Kim et al. 2011] also provide a powerful approach. However, these previous algorithms still do not treat the matching step as an independent problem, even if matching is not tightly coupled with feature description and selection. This paper focuses on the feature matching problem as a 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 by [Conte et al. 2004], matching single features leads to a linear assignment problem, but if multiple features are matched at once, 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 the distances between the points in each tuple being matched. This helps to reject many false matches, making the results more reliable in terms of greatly improving the matching accuracy. 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 [Leordeanu and Hebert 2005] consider pairs of feature descriptors, and use 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 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, used 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 of the triangles generated by such triples. However, 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 [Kofidis and Regalia 2002].

Our *SuperMatching* algorithm also formulates higher-order matching problems using a supersymmetric affinity tensor. It can accurately match a moderate number 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 geometric 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 devise 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, while having a similar computational cost. Importantly, the matching approach is general as it is independent of choice of 2D or 3D feature descriptors, feature point selection method, and constraints placed on tuples.

2 Related work

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 features, one from each set being matched. Affinity measures used in computer graphics and computer vision tasks are defined typically as the feature distance between feature vectors based on local information around each feature point, e.g. SIFT [Lowe 2004], spin images [Johnson and Hebert 1999], slippage features [Bokeloh et al. 2008], heat diffusion signatures [Sun et al. 2009], and BRISK [Leutenegger et al. 2011]. Point-to-point matching can give misleading results as wrong correspondences are 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 distance between the points in a pair, assuming the objects are related by a rigid transformation [Leordeanu and Hebert 2005; Cour et al. 2006], or geodesic distance, assuming 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]. 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, and various approximate methods have been developed. [Zass and Shashua 2008] 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. [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 and Keller 2010] treated 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. 2010] also built a multiple higher-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 match moderate numbers (many hundreds) of features. However, these 2D approaches do not take advantage of supersymmetry of the affinity tensor, SuperMatching does so, leading to an improvement in matching accuracy. 3D problems are even more challenging.

A related idea using higher order constraints in 3D registration, the 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 need to find coplanar 4-tuples of points and the assumption of rigid transformation limit its applicability. We solve both rigid and isometric shape matching problems with a single approach.

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 [Kolda and Bader 2009]. More formally, an Nth-order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system.

Assume we are given two sets of feature points P and Q, with N_1 and N_2 points respectively. From the Nth-order tensor viewpoint, matching between these two feature sets can be represented by an *assignment variable* $\boldsymbol{x}$. The matching problem is equivalent to finding the optimal assignment tensor $\boldsymbol{x}^* =< x_{i_1}, \cdots, x_{i_N} > \in \{0,1\}^N$, satisfying [Kolda and Bader 2009; Duchenne et al. 2009]

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

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 and Q be F_1 and F_2, then $\forall(f^1_{i_1}, \cdots, f^1_{i_N}) \in F_1$, there is a corresponding 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 otherwise 0. $\mathcal{T}_N(i_1, \cdots, i_N)$ is the affinity of the set of assignments $\{i_n\}_{n=1}^N$, which will be high if the tuple of features $(f^1_{i_1}, \cdots, f^1_{i_N})$ is similar to the tuple $(f^2_{i_1}, \cdots, f^2_{i_N})$, and distances are similar. Note that the size of $\mathcal{T}_N(i_1, \cdots, i_N)$ is $(N_1 N_2)^N$. In this paper, the affinity measures expressing similarity of feature tuples are stored using a supersymmetric tensor which is at the basis of our SuperMatching algorithm.

In the rest of the paper, we consider the one-to-many correspondence problem. We assume that each point in P is matched to exactly one point in Q, 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 to Q, then match Q to P, and then combine the matching results by taking their union or intersection.

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 $\mathcal{T}_N$ in a supersymmetric manner? (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? (see Section 4.3)
- 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 4.4)

4 SuperMatching

We now discuss the first two 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

Here we consider the supersymmetric higher-order affinity tensor, which 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 [Kofidis and Regalia 2002].*

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

Definition 2 (Supersymmetric Affinity Tensor) *Given two feature sets P and Q, with N_1 and N_2 features respectively, the supersymmetric affinity tensor is an Nth order $I_1, \cdots, I_N$, nonnegative tensor $\mathcal{T}_N$, for which there exists a set of indices θ_N, and an Nth order potential function ϕ_N, such that*

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

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

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

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

1: Initialize $\boldsymbol{u}_0$ by randomly positive values, $k = 1$
2: **repeat**
3: **for** all $(i_1, i_2, \cdots, i_N) \in \theta_N$ **do**
4: **for** all $m \in (i_1, \cdots, i_N)$ **do**
5: $v_m^{(k)} = (N-1)!\phi_N(i_1, \cdots, i_N) 2 v_m^{(k-1)} v_{i_1}^{2(k-1)} \cdots v_{m-1}^{2(k-1)} v_{m+1}^{2(k-1)} \cdots v_{i_N}^{2(k-1)}$
6: **end**
7: **for** $i = 1 : N_1$ **do**
8: $v^{(k)}(((i-1)\cdot N_2 + 1) : i \cdot N_2) = \hat{v}^{(k)}(((i-1)\cdot N_2 + 1) : i \cdot N_2) / \|\hat{v}^{(k)}(((i-1)\cdot N_2 + 1) : i \cdot N_2)\|_1$
9: **end**
10: **end**
11: $k = k + 1$;
12: **until** convergence;

Note: $\boldsymbol{u}^{(k)} = \boldsymbol{v}^{2(k)}$, ϕ_N is the corresponding potential function, 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 $i \cdot N_2$.

A tensor element with $(i_1, \ldots, i_N) \in \theta_N$ is called a *potential element*, while other elements are called *non-potential element*. A potential element is one real matching result out of all possible matching candidates. The potential elements would be further detailed in the Section 4.4.

Using Definition 2, we now can reduce the amount of storage needed, representing every potential element $\mathcal{T}_N(i_1, \ldots, i_N)$ by the canonical entry $\mathcal{T}_N(\text{sort}(i_1, \ldots, i_N))$, $\forall (i_1, \ldots, i_N) \in \theta_N$. Each stored value thus provides the value for $N!$ entries. Furthermore, as non-potential elements all have value 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 4.4. At the same time, it can be used to make the power iteration process more efficient: see Section 4.2.

4.2 Supersymmetric Higher-order Power Iteration

The higher-order tensor problem in Eq. (1) may be solved by the tensor decomposition method [Kolda and Bader 2009]. Tensor decomposition originated in [Hitchcock 1927]. We utilize the rank-one higher-order power method [Lathauwer et al. 1995] to approximately solve the Equ.(1); as noted, an exact computation is infeasible. So, Equ.(1) can be expressed as:

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

where $\star$ is called the Tucker product [Kofidis and Regalia 2002], and $\boldsymbol{x} \in \{0, 1\}^N$. 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 elements taking real values in $[0, 1]$. This changes the optimization problem to one of computing the rank-one approximation of the affinity tensor $\mathcal{T}_N$ [Kofidis and Regalia 2002], 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 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 (or concavity) for the functional induced by the tensor [Kofidis and Regalia 2002], 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 [Duchenne et al. 2009] uses the ℓ^1 norm to replace the traditional ℓ^2 norm.

We both use the ℓ^1 norm, and further revise S-HOPM as follows. To perform higher-order power iteration of $\boldsymbol{u}$, 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 the unit tensor [Kofidis and Regalia 2002]. For $\hat{\boldsymbol{u}}^{(k)}$ belonging to an Nth-order supersymmetric affinity tensor, this can be formulated as follows:

$$\hat{\boldsymbol{u}}^{(k)} = \mathcal{I} \overset{\mathcal{T}_N}{\star} (\boldsymbol{u}^{(k-1)}) \overset{\mathcal{T}_N}{\star} {}^{(N-1)} \text{ implies that } \forall m \in (i_1, ..., i_N),$$

$$v_m^{(k)} = \sum_{i_1,...,i_N} \mathcal{T}_N(i_1, ..., i_N) 2 v_m^{(k-1)} v_{i_1}^{2(k-1)} ... v_{m-1}^{2(k-1)} v_{m+1}^{2(k-1)} ... v_{i_N}^{2(k-1)} =$$

$$(N-1)! \phi_N(i_1, ..., i_N) 2 v_m^{(k-1)} v_{i_1}^{2(k-1)} ... v_{m-1}^{2(k-1)} v_{m+1}^{2(k-1)} ... v_{i_N}^{2(k-1)} \quad (4)$$

where $\boldsymbol{u}^{(k)} = \boldsymbol{v}^{2(k)}$, and ϕ_N is the corresponding potential function that would be detailed in the following Section 4.3.

Note that Eq. (4) is more compact than earlier expressions in the literature, as it handles all symmetrically related potential elements as a single item using multiplication by $(N-1)!$. The efficiency of our SuperMatching algorithm relies on two principles. First, we take advantage of the supersymmetry to deduce $\boldsymbol{u}$ as in Equ.(4). Secondly, many of the elements of the affinity tensor are zero non-potential elements, and it is much more efficient to perform the power iteration by just considering the non-zero potential elements.

Our supersymmetric higher-order power iteration solution of Eq. (1) is performed by the SuperMatching algorithm—See Algorithm 1. It excludes each non-potential element from the iteration process and only utilizes one canonical potential element for each related element, so it is efficient. The complexity of the whole iteration process only depends on the number $|\theta_N|$ of non-zero affinities. Step 5 in Algorithm 1 represents all permutations of each potential element $\mathcal{T}_N(i_1, \cdots, i_N)$ using a single potential function $\phi_N(i_1, \cdots, i_N)$. Consequently, this method reduces memory costs while keeping accuracy. Note that, although [Duchenne et al. 2009] claimed to use a supersymmetric affinity tensor, his approach does not make full use of supersymmetry when creating the supersymmetric affinity tensor, nor does it take advantage of supersymmetry to accelerate the power iteration process. By doing so, we overcome limitations due to unbalanced and redundant tensor elements in [Duchenne et al. 2009], as our experiments show later.

Many initialization schemes have been proposed for the S-HOPM method [Kofidis and Regalia 2002]. We simply use positive random values between 0 and 1 to initialize $\boldsymbol{u}_0$, which ensures convergence; proofs are detailed in [Regalia and Kofidis 2000; Kofidis and Regalia 2002].

4.3 Higher-order Potentials

Different higher-order potentials are appropriate for different applications. Here we give two general higher-order potentials. One may be used for 2D cases, while the other is useful for 3D matching. 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.

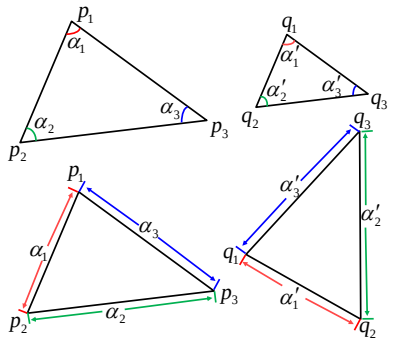

Figure 2: *Third-order potential. The geometric constraints are: internal angle invariance in 2D (above), and edge length invariance in 3D (below).*

In 2D, we first restate a well-known 2D third-order geometric-similarity invariant potential ϕ_3 [Duchenne et al. 2009; Chertok and Keller 2010] for linking two point feature triples. Similarity of triangles formed by three points corresponds to invariance under scaling, rotation and translation—interior angles do not change. Thus ϕ_3 can be defined in terms of differences of corresponding interior angles:

$$\begin{aligned} \phi_3(i_1, i_2, i_3) &= \phi_3(\{p_1, q_1\}, \{p_2, q_2\}, \{p_3, q_3\}) \\ &= \exp(-1/\varepsilon^2 \sum\nolimits_{(l,l')} \|\alpha_l - \alpha_{l'}\|^2) \end{aligned} \quad (5)$$

where $\varepsilon > 0$ is the is the kernel bandwidth, $\{\alpha_l\}_{l=1}^3$ and $\{\alpha_l^{'}\}_{l'=1'}^3$ 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. We may extend it to the general case by using the internal angles formed by polygons with more than 3 sides. It is easy to see that the potential preserves invariance under rigid transformations in 2D field.

For 3D matching problems, we may replace the internal angle by edge length, i.e. the geodesic distance across the mesh in which the points are embedded. which now corresponds to assuming an isometry transform relating the point sets. Geodesic distance may be computed by Dijkstra's algorithm [Peyré et al. 2010] .

We will use these two high-order potentials to evaluate our algorithm. Figure 2 illustrates the third-order potential examples in the 2D and 3D cases.

4.4 Sampling Strategy

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

For the two feature sets P and Q, 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 and Q, as F_1 and F_2; then $\forall (f^1_{i_1}, \cdots, f^1_{i_N}) \in F_1$, calculate the potentials for $(f^1_{i_1}, \cdots, f^1_{i_N})$ with all feature tuples in F_2. This naive method is very time-consuming, which is why sampling is used. We employ random sampling for general feature matching problems, but this does not preclude more directed sampling if prior knowledge of the matching problems gives guidance.

Our sampling approach is to repeatedly randomly sample t_1 feature tuples for each feature point from P, and fully sample Q. For P, 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 have been chosen once as a required element. So the number of feature tuples in F_1 is $N_1 t_1$, and N_2^N for F_2. Then, $\forall (f_{i_1}^1, \cdots, f_{i_N}^1) \in F_1$, we find k most similar features in F_2 to build N 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, the sampling cost is $O(N_1\, t_1\, k\, log N_2)$. 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

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

where Ω is an arbitrary permutation.

Earlier work [Zass and Shashua 2008; Duchenne et al. 2009; Wang et al. 2010] 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 $(f_{i_1}^1, f_{i_2}^1, f_{i_3}^1)$ may be sampled from P and $(f_{i_1}^2, f_{i_2}^2, f_{i_3}^2)$ from Q, and also a feature tuple $(f_{i_1}^1, f_{i_3}^1, f_{i_2}^1)$ sampled from P and $(f_{i_1}^2, f_{i_3}^2, f_{i_2}^2)$ from Q. 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 $(s_{i_1}, s_{i_2}, s_{i_3})$ for any permutation of indices. This 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 unbalances the power iteration process, and can lead to inaccurate results. Therefore, our sampling method reduces the sampling cost, while also improving the accuracy of the power iteration.

5 Experiments

We have used synthetically generated data as well as real captured data to evaluate the SuperMatching algorithm. To demonstrate that the SuperMatching algorithm is general (independent of feature descriptors), several descriptors have been used. For input shapes without color information, a uniform sampling of points on the source shape were employed. For colored shapes, both SIFT feature points and uniform sampling points were used. We used third-order matching in our experiments, and note it would be simple to use higher order.

5.1 3D Rigid Shapes Scans

Firstly, we used SuperMatching to align 3D rigid shape scans pairwise based on the uniform sampling scheme. Rigid transforms can be computed from three compatible sampling points. The transform which brings the most data points within a threshold of a point in the model is chosen as the optimal aligning transform [Huttenlocher and Ullman 1990]. As discussed in [Gelfand et al. 2005], such a voting scheme is guaranteed to find the optimal alignment between the pairwise scans and is independent of the initial pose of the input scans. Figure 3 shows some registration results for the Rooster model from [Chuang et al. 2009]. On the left column is the original state, then the following columns are our matching and registered result (without any ICP refinement [Besl and McKay 1992]); on the right column is the result produced by [Aiger et al. 2008].

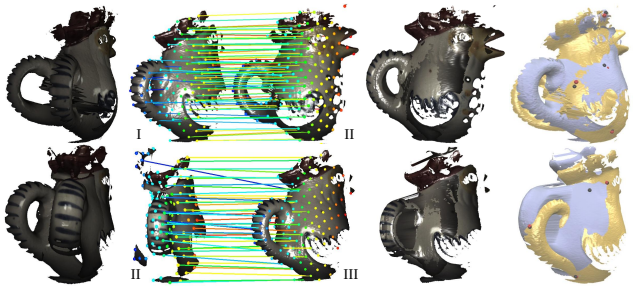

Figure 3: *Pairwise alignment of Rooster $I - II$ and $II - III$ scans. Left : our result. Right: result from [Aiger et al. 2008].*

Next, we extended the 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 the initial pairwise matching, the alignment was refined by the iterative closest point (ICP) algorithm [Besl and McKay 1992]. Figure 4 illustrates the approach. Above, a Rooster is aligned by our method, and the top-right is the real ground truth [Chuang et al. 2009]. Below, matching is used to align 8 partial scans which are then merged to produce a single shape. Pairs of consecutive scans, linked by dark lines, are matched using the SuperMatching algorithm.

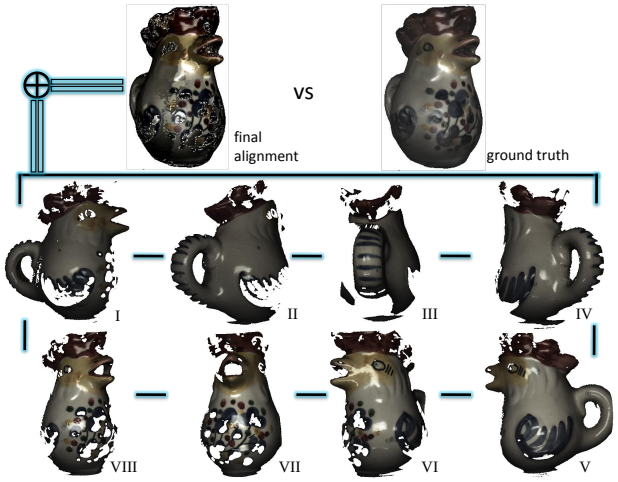

Figure 4: *Alignment of several Rooster scans from different viewpoints. Above: our final registered Rooster vs the ground truth [Chuang et al. 2009]. Below: 8 partial scans, the dark lines indicating the pairwise matches.*

5.2 3D Depth Scans with Color Information

We next provide a real-world noisy example of the use of SuperMatching. In this case, real world data with surface color information was captured using a Kinect camera [Kinect 2012], and both

SIFT and uniform sampling points were used as a basis for SuperMatching. This resulted in robust matches without significant outliers, as illustrated in Figure 5. The example also demonstrates that the SuperMatching is general, in the sense that it is independent of feature descriptors.

Figure 5: *3D real depth scans with color information, captured using Kinect. Above: two given different local pre-scans. Below: a single scan. Matching points are connected by green lines.*

5.3 3D Articulated Shape Synthetic Data

Thirdly, we present another application, registration of (approximately) articulated shapes. Such problems are common in dynamic range scanning. Given a sequence of range scans of a moving articulated subject, our method automatically registers all data to produce a complete 3D shape. Note that, unlike many other methods, our method does not need manual segmentation, markers, or a prior template. While the problem of non-rigid registration of deformable shapes is ill-posed and no algorithm is applicable to all scenarios, we believe that our approach pushes the limits of what can be achieved with minimal prior information, and is robust given partial data with holes.

When doing pairwise articulated shape registration, although the partial scans have missing data and their poses are different, SuperMatching still produces accurate matching. Correspondences on uniform sampling points on the two shapes are established by SuperMatching; these permit robust registration of scans by computing piecewise rigid transformations. These transformations are propagated from the slippage feature points to the entire set of points in each scan using nearest neighbor interpolation. Figure 6 shows two registration examples of an articulated model. On the left is our result, on the right is the registered result produced by the method in [Chang and Zwicker 2009].

For a sequence of partial articulated data, the registration is performed in two main steps. We first precompute an initial pairwise registration for each pair of consecutive frames, then perform articulated shape reconstruction as in [Pekelny and Gotsman 2008]. Segmentation of the scans into rigid parts can readily be done by clustering the transformations obtained from the slippage feature points, using the mean shift algorithm [Comaniciu and Meer 2002]. This information is used as the input to the second step

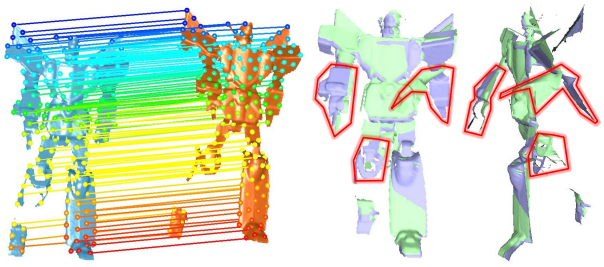

Figure 6: *Pairwise matching of articulated Robot between frame 9 and 10. Left: our result. Right: result produced by [Chang and Zwicker 2009] from front and side views, where red polygons indicate the large distortion regions.*

of articulated shape reconstruction following [Pekelny and Gotsman 2008]; this algorithm identifies and tracks the rigid parts in each frame, while accumulating geometric information over time. However, [Pekelny and Gotsman 2008] requires the user to manually segment each range scan in advance, whereas we automatically determine the segmentation. Figure 7 shows an articulated hand example. This synthetic data is generated from a deformation sequence, and the final registered shape is produced from these partial data. By using synthetic data, we are able to evaluate the robustness of our reconstruction method using the ground truth, as shown in Figure 7. Quantitatively, we measured the maximum of the average distance of the reconstruction over all frames as $0.001D$ where D is the bounding box diagonal length, and the greatest distance error in any one frame was $0.012D$.

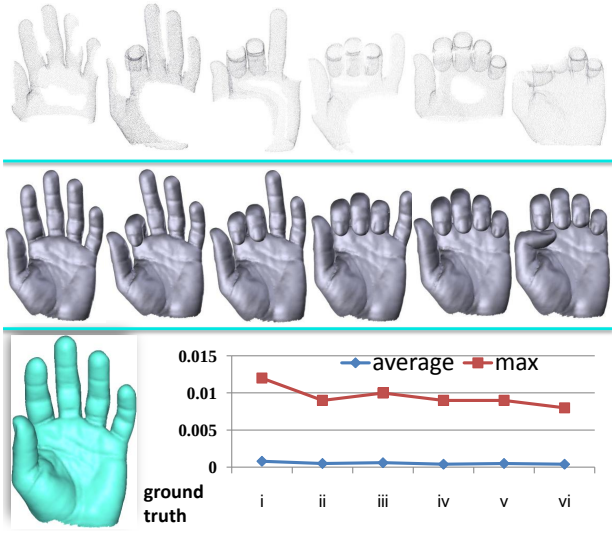

Figure 7: *Registration of an articulated hand. Above: partial synthetic data with holes is generated from a deformation sequence. The reconstructed meshes are deduced from the registration process (center). Below: first frame ground truth shape, and average and maximum distance from the ground truth per frame.*

5.4 Deformable Surfaces

Finally, we 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 in-

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

Table 1: *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 (s)
SuperMatching	83%	85%	84%	81%	66%	60%	69%	56%	8
[Zass and Shashua 2008]	73%	79%	70%	72%	44%	39%	54%	43%	6.5
[Duchenne et al. 2009]	67%	77%	73%	65%	39%	31%	47%	42%	13
[Cour et al. 2006]	27%	29%	22%	27%	14%	5%	28%	7%	5

cluded 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 the spectral algorithm [Cour et al. 2006] (a quadratic assignment algorithm), a third-order tensor algorithm [Duchenne et al. 2009], and the hyper graph matching algorithm [Zass and Shashua 2008], using the authors' code in each case. All methods were executed in Matlab on a 2.3GHz Core2Duo with 2GB memory. To enable direct and fair comparison, [Duchenne et al. 2009], [Zass and Shashua 2008] and SuperMatching used the same potential and all used the same tensor size $(N_1 N_2)^N$.

In the tests, SuperMatching considered 3×10^6 feature tuples, while the method of [Duchenne et al. 2009] considered 10×10^6 features and the method of [Zass and Shashua 2008] used 4×10^6. The difference mainly results from differences in sampling strategy; note that we have the lowest sampling cost. The average running time to match two feature sets each with 100 features was around 8s for SuperMatching, 13s for [Duchenne et al. 2009], 6.5s for [Zass and Shashua 2008], and 5s for [Cour et al. 2006]. SuperMatching takes less time than the third-order tensor algorithm in [Duchenne et al. 2009] as it uses the same tensor size but fewer feature tuples.

Matching accuracy is 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 1 and illustrated in Figure 8. Table 1 demonstrates that SuperMatching achieves a higher matching accuracy than previous algorithms. The worst matching result is produced by the spectral quadratic assignment algorithm [Cour et al. 2006], due to the lower discriminatory power of the pairwise geometric constraints used. Higher-order algorithms perform better due to the more complex geometric constraints. Nevertheless, SupeMatching also significantly outperforms the third-order algorithm [Duchenne et al. 2009] and the hyper graph matching algorithm [Zass and Shashua 2008], as these do not tale proper advantage of supersymmetry.

6 Conclusion

This paper has presented the novel 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, whose efficiency 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.

Acknowledgements. We are grateful to them for sharing related source codes, execution programs, and testing data, whose name would appear after the review. Then SuperMatching source codes would be public under the GNU Lesser General Public License.

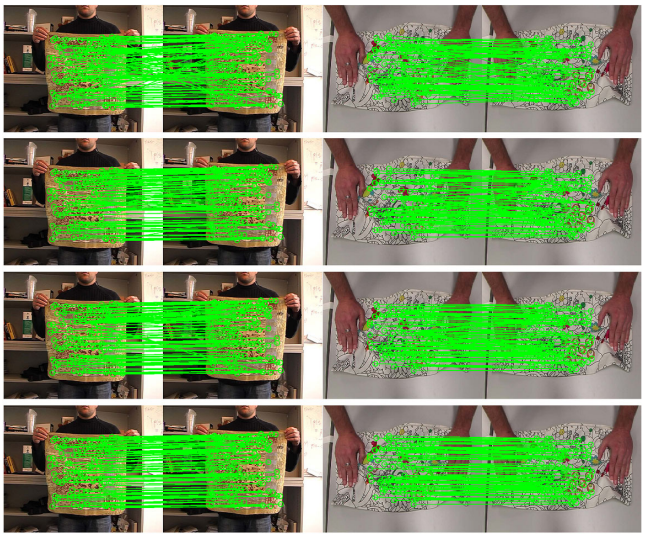

Figure 8: *Matching results. Left: cloth set, matching between frame 80 and 90, right: cushion set, matching between 144 and 156. Top to bottom, spectral method [Cour et al. 2006], hyper graph matching method [Zass and Shashua 2008], a Third-order tensor [Duchenne et al. 2009], and SuperMatching algorithm.*

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.

BERG, A. C., BERG, T. L., AND MALIK, J. 2005. Shape matching and object recognition using low distortion correspondence. In *IEEE CVPR*, 26–33.

BESL, P. J., AND MCKAY, N. D. 1992. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 239–256.

BOKELOH, M., BERNER, A., WAND, M., SEIDEL, H.-P., AND SCHILLING, A., 2008. Slippage features. Technical Report, WSI-2008-03, University of Tübingen,.

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 26*.

CHANG, W., AND ZWICKER, M. 2009. Range scan registration using reduced deformable models. *Computer Graphics Forum (Proc. Eurographics) 28*, 2, 447–456.

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.

CHUANG, M., LUO, L., BROWN, B. J., RUSINKIEWICZ, S., AND KAZHDAN, M. M. 2009. Estimating the laplace-beltrami operator by restricting 3d functions. *Computer Graphics Forum (Proc. SGP) 28*, 5, 1475–1484.

COMANICIU, D., AND MEER, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 603–619.

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.

COUR, T., SRINIVASAN, P., AND SHI, J. 2006. Balanced graph matching. In *NIPS*, 313–320.

DUCHENNE, O., BACH, F., KWEON, I., AND PONCE, J. 2009. A tensor-based algorithm for high-order graph matching. In *IEEE CVPR*, 1980–1987.

GELFAND, N., MITRA, N. J., GUIBAS, L. J., AND POTTMANN, H. 2005. Robust global registration. In *Symposium on Geometry processing*.

HITCHCOCK, F. L. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics 6*, 064–089.

HUTTENLOCHER, D. P., AND ULLMAN, S. 1990. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision 5* (November), 195–212.

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.

KIM, V. G., LIPMAN, Y., AND FUNKHOUSER, T. 2011. Blended intrinsic maps. In *SIGGRAPH*, 79:1–79:12.

KINECT, 2012. Kinect homepage. http://www.xbox.com/en-US/kinect.

KOFIDIS, E., AND REGALIA, P. A. 2002. On the best rank-1 approximation of higher-order supersymmetric tensors. *SIAM Journal on Matrix Analysis and Applications 23*, 3, 863–884.

KOLDA, T. G., AND BADER, B. W. 2009. Tensor decompositions and applications. *SIAM Review 51*, 3, 455–500.

LATHAUWER, L. D., COMON, P., MOOR, B. D., AND VANDEWALLE, J. 1995. Higher-order power method. In *Proceedings of NOLTA*, 2709–2712.

LEORDEANU, M., AND HEBERT, M. 2005. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision*, 1482–1489.

LEUTENEGGER, S., CHLI, M., AND SIEGWART, R. 2011. Brisk: Binary robust invariant scalable keypoints. In *International Conference of Computer Vision*.

LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum (Proc. SGP) 27*, 5, 1421–1430.

LIPMAN, Y., AND FUNKHOUSER, T. 2009. Möbius voting for surface correspondence. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 72:1–72:12.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 91–110.

OVSJANIKOV, M., MRIGOT, Q., MMOLI, F., AND GUIBAS, L. 2010. One point isometric matching with the heat kernel. *Computer Graphics Forum (Proc. SGP) 29*, 5, 1555–1564.

PEKELNY, Y., AND GOTSMAN, C. 2008. Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum (Proc. EuroGraphics) 27*, 2, 399–408.

PEYRÉ, G., PÉCHAUD, M., KERIVEN, R., AND COHEN, L. D. 2010. Geodesic methods in computer vision and graphics. *Foundations and Trends in Computer Graphics and Vision 5*, 197–397.

REGALIA, P. A., AND KOFIDIS, E. 2000. The higher-order power method revisited: convergence proofs and effective initialization. In *Proceedings of the Acoustics Speech and Signal Processing*, IEEE Computer Society, 2709–2712.

SAHILLIOGLU, Y., AND YEMEZ, Y. 2011. Coarse-to-fine combinatorial matching for dense isometric shape correspondence. *Computer Graphics Forum (Proc. SGP) 30*, 5, 1461–1470.

SUN, J., OVSJANIKOV, M., AND GUIBAS, L. 2009. A concise and provably informative multi-scale signature based on heat diffusion. In *Symposium on Geometry Processing*, 1383–1392.

SUN, J., CHEN, X., AND FUNKHOUSER, T. A. 2010. Fuzzy geodesics and consistent sparse correspondences for deformable shapes. *Computer Graphics Forum 29*, 5, 1535–1544.

TEVS, A., BOKELOH, M., WAND, M., SCHILLING, A., AND SEIDEL, H.-P. 2009. Isometric registration of ambiguous and partial data. In *IEEE CVPR*, 1185–1192.

TEVS, A., BERNER, A., WAND, M., IHRKE, I., AND SEIDEL, H.-P. 2011. Intrinsic shape matching by planned landmark sampling. *Computer Graphics Forum 30*, 2, 543–552.

TOLER-FRANKLIN, C., BROWN, B., WEYRICH, T., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2010. Multi-feature matching of fresco fragments. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) 29*, 185:1–185:12.

TORRESANI, L., KOLMOGOROV, V., AND ROTHER, C. 2008. Feature Correspondence Via Graph Matching: Models and Global Optimization. In *the 10th European Conference on Computer Vision*, Springer-Verlag, 596–609.

VAN KAICK, O., ZHANG, H., HAMARNEH, G., AND COHEN-OR, D. 2011. A survey on shape correspondence. *Computer Graphics Forum 30*, 6, 1681–1707.

WAND, M., ADAMS, B., OVSJANIKOV, M., BERNER, A., BOKELOH, M., JENKE, P., GUIBAS, L., SEIDEL, H.-P., AND SCHILLING, A. 2009. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics 28*, 15:1–15:15.

WANG, A., LI, S., AND ZENG, L. 2010. Multiple order graph matching. In *Asian Conference on Computer Vision*, 471–482.

WINDHEUSER, T., SCHLICKWEI, U., SCHIMDT, F. R., AND CREMERS, D. 2011. Large-scale integer linear programming for orientation preserving 3d shape matching. *Computer Graphics Forum (Proc. SGP) 30*, 5, 1471–1480.

ZASS, R., AND SHASHUA, A. 2008. Probabilistic graph and hypergraph matching. In *IEEE CVPR*, 1–8.

ZENG, Y., WANG, C., WANG, Y., GU, X., SAMARAS, D., AND PARAGIOS, N. 2010. Dense non-rigid surface registration using high-order graph matching. In *IEEE CVPR*, 382–389.