2013 IEEE International Conference on Computer Vision

A Fully Hierarchical Approach for Finding Correspondences in Non-rigid Shapes

Ivan Sipiran[1] and Benjamin Bustos[2]

[1]Department of Computer and Information Science, University of Konstanz
[2]Department of Computer Science, University of Chile

Abstract

This paper presents a hierarchical method for finding correspondences in non-rigid shapes. We propose a new representation for 3D meshes: the decomposition tree. This structure characterizes the recursive decomposition process of a mesh into regions of interest and keypoints. The internal nodes contain regions of interest (which may be recursively decomposed) and the leaf nodes contain the keypoints to be matched. We also propose a hierarchical matching algorithm that performs in a level-wise manner. The matching process is guided by the similarity between regions in high levels of the tree, until reaching the keypoints stored in the leaves. This allows us to reduce the search space of correspondences, making also the matching process efficient. We evaluate the effectiveness of our approach using the SHREC'2010 robust correspondence benchmark. In addition, we show that our results outperform the state of the art.

1. Introduction

The problem of finding correspondences in 3D shapes is an important problem in computer vision and computer graphics. In particular, the reliable detection of correspondences in non-rigid shapes has received notable attention in recent years. This problem is difficult to solve due to the complexity of formally characterizing a non-rigid transformation. Additionally, in real-world applications, one would expect shapes containing perturbations such as noise, topological changes, scale, and so on. Therefore, it is imperative to devise robust and efficient techniques to finding reliable correspondences in non-rigid shapes (possibly with perturbations).

The most used approach to tackle this problem involves finding a mapping between sparse sets of surface points (keypoints). This problem commonly is formulated as an optimization problem that involves the matching of local descriptors and some criterion for geometric consistency. Generally, the optimization is carried out through an integer quadratic program.

Due to the combinatorial nature of the correspondence problem, we need to search strategies to efficiently solve the problem. Additionally, we need to take into account the robustness against mesh perturbations. In this paper, we propose an algorithm to find correspondences in non-rigid shapes based on a hierarchical decomposition. Recent studies have shown that it is possible to obtain robust decompositions on 3D meshes [9, 15]. Our motivation is the fact that if two shapes are near-isometric, they should have regions which are near-isometric as well. If a shape S is decomposed in a set of regions $S_1, S_2, \ldots, S_n$, then a shape T (near-isometric to S) should also have a partition set $T_1, T_2, \ldots, T_n$, where S_i is near-isometric to T_j. This idea can be applied again on regions recursively.

In light of this observation, we propose an algorithm to build a *decomposition tree* of a given shape. Internal nodes represent regions and leaf nodes represent keypoints. As one traverses the tree in depth, the shape structures are smaller. In addition, we propose a hierarchical matching algorithm which takes two decomposition trees as input and performs in a breadth-first manner. This algorithm first matches regions in high levels of the trees and propagates the process until reaching the leaf nodes, where the keypoints are finally matched. The decomposition tree is similar in spirit to the component tree proposed by Litman *et al*. [9]. The main difference is that our representation is designed to support the subsequent matching process. In addition, in contrast to the component tree (which makes use of diffusion geometry), our method uses the distribution of keypoints on the surface to guide the decomposition process.

In our method, we address two important aspects: efficiency and robustness. Regarding efficiency, our matching

1550-5499/13 $31.00 © 2013 IEEE
DOI 10.1109/ICCV.2013.106

algorithm has the ability to reduce the searching space of correspondences by making use of the hierarchical decomposition. Once two regions correspond (because their internal nodes matched), we only look for correspondences in the associated sub-trees. This allows us to discard a considerable amount of matches early in the process. With respect to the robustness, we take advantage of the provably good properties of diffusion-based descriptors in the context of non-rigid matching to obtain discriminative descriptions for regions and keypoints. In addition, the decomposition process is guided by the distribution of robust keypoints on the shape's surface.

In summary, the contributions of our work are twofold. First, we present a novel representation for non-rigid shapes: the *decomposition tree*. This structure characterizes the hierarchical decomposition process, where the root node contains the original shape and leaf nodes contain keypoints. Second, we develop a matching algorithm that takes advantage of the decomposition trees to find correspondences. Our algorithm is efficient since it allows us to discard matches in early stages of the process.

Our paper is organized as follows. Section 2 presents the related work. Section 3 describes our hierarchical decomposition algorithm and the related decomposition tree. Section 4 is devoted to present the matching algorithm. Section 5 presents our experiments and discusses the obtained results. Finally, Section 6 draws the conclusions of our work and gives directions for future works.

2. Related Works

The matching problem can be formulated as a minimum-distortion problem. Given two near-isometric shapes, the goal is to find a correspondence set that minimizes the distortion between the two shapes. Generally, this problem is stated as an optimization problem where solutions arise from the need to resolve this problem effectively and efficiently.

Bronstein *et al*. [5] proposed to match two shapes by embedding one into another. Shapes were treated as metric spaces using the geodesic distance as metric. They used a generalization of the multi-dimensional scaling method to find the minimum-distortion embedding between two metric spaces. On the other hand, a Möbius voting approach was used by Lipman and Funkhouser [8]. A conformal flattening was applied to shapes, where it was possible to apply a Möbius transform for triplets of points. The deformation energy caused by the triplets in the rest of the points of the mesh allows us to compute a vote for each couple of correspondences. The pairs with high votes were considered as reliable correspondences.

Many methods solve the problem iteratively through a process of correspondence refinement. Ovsjanikov *et al*. [10] exploited the use of heat kernel maps to find a unique reliable correspondence. Subsequently, an iterative method propagates the correspondence set according to a kernel map optimization method. Also, a RANSAC-like method to find correspondences was proposed in [18]. The algorithm operates over a set of hypothesis from an initial set computed from keypoints. Similarly, Tevs *et al*. [17] presented a probabilistic approach to plan the iterative searching of correspondences.

Greedy approaches to tackle the optimization problem have been also considered. Zhang *et al*. [21] selected keypoints using a geodesic approach. Then, the search of correspondences was guided by a best-first algorithm that models the quality of the correspondence set with a geodesic distortion. On the other hand, Zaharescu *et al*. [19] proposed a method which resembles the matching of correspondences in images. Adaptations of DoG and HoG were used for selecting and describing keypoints on the mesh surface. Next, a simple matching algorithm that takes the best match between descriptors was presented. Also, Sahillioğlu and Yemez [12] stated the correspondence problem as a graph problem. They proposed to find an initial matching over a bipartite graph, and subsequently a greedy approach was responsible of adding and refining new correspondences. Similarly, Sharma *et al*. [13] proposed a probabilistic method to find dense correspondences based on a initial greedy-based sparse matches.

A common approach is to state the problem of finding correspondences as an integer program. Dubrovina and Kimmel [6] described a mesh point using the eigenfunctions of the Laplace-Beltrami operator. The matching algorithm was stated as a quadratic integer program which takes into account a distortion function to optimize. More recently, Rodolà *et al*. [11] formulated the minimum-distortion problem with a quadratic optimization function. The solution was obtained with a game-theoretic strategy and a merging algorithm to get the final correspondence set. Zeng *et al*. [20] devised a high-order matching algorithm based on the Möbius transform to populate the correspondences from an initial set. They derived a dual decomposition of the original problem which involved an integer linear problem.

3. The Decomposition Tree

In this section, we present the decomposition algorithm. It has two stages: the pre-processing step and the generation of the decomposition tree. Our approach to decompose a mesh is inspired in the our method to detect key-components on shapes [15].

3.1. Pre-processing

Given a 3D shape X, we compute a set of keypoints K_X using the Harris 3D algorithm [14]. Subsequently, we calculate the geodesic distances between each pair of keypoints using the fast marching method [7]. These distances will be

used to control the geometric consistency of the correspondence set in the matching algorithm.

Since the decomposition method is based on the distribution of keypoints on the shape's surface, we propose a heuristic to discard isolated keypoints. The heuristic consists of analyzing the distribution of geodesic distances of a keypoint. Let D_p^v be the geodesic distances from a keypoint v to its p nearest keypoints. We define the *density* of a keypoint v as

$$density(v) = \frac{mean(D_n^v)}{mean(D_3^v)}, \tag{1}$$

where $n = |K_X|$.

Grouped keypoints are expected to have a small $mean(D_3^v)$ compared to $mean(D_n^v)$, yielding a high density value. In contrast, isolated keypoints will have more similar values for these two quantities, with the density approaching to one. Therefore, keypoints with a low density value should be discarded. We remove keypoints with $density(v) < 30$ from K_X. The cut-off value was found empirically.

In addition, for each remaining keypoint in K_X, our method computes two descriptors which are based on diffusion geometry: HKS (heat kernel signatures [16]) and WKS (wave kernel signatures [1]). Both descriptors are computed from the eigenfunctions of the Laplace-Beltrami operator of the shape. The HKS of a point x is defined as

$$HKS(x) = (hks_{t_1}(x,x), \ldots, hks_{t_n}(x,x)), \tag{2}$$

where

$$hks_t(x,x) = \sum_{k \geq 1} exp(-\lambda_k t)\phi_k^2(x), \tag{3}$$

and where λ_k and $\phi_k(\cdot)$ are respectively the eigenvalues and eigenfunctions of the Laplace-Beltrami operator. Similarly, the WKS is defined as

$$WKS(x) = (wks_{e_1}(x), \ldots, wks_{e_n}(x)), \tag{4}$$

where

$$wks_e(x) = \sum_{k \geq 1} f_e^2(\lambda_k)\phi_k^2(x), \tag{5}$$

and where f_e is a family of log-normal energy distributions.

The use of these two descriptors is related to their ability for feature localization [3]. The heat kernel signature is a collection of low-pass filters which inhibit high frequencies. For this reason, its nature is more global and it may affect the exact localization of correspondences. In contrast, the wave kernel signature is a collection of band-pass filters which reduces the impact of low frequencies, allowing a better feature localization. Therefore, we take advantage of these facts and use the HKS and WKS in different levels of our representation. The former is used for describing regions in the internal nodes of our decomposition tree and the latter is used for describing keypoints in the leaf nodes.

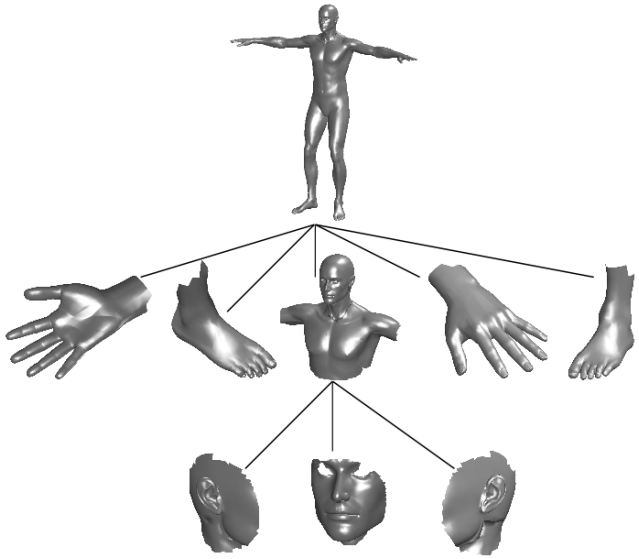

Figure 1. A decomposition tree of a human shape. The root node contains the original mesh and its related data. The first decomposition generates five regions. The last level contains smaller regions from the head.

In summary, after the pre-processing step, we have the following information: a set of keypoints K_X, the complete set of geodesic distances between keypoints, and descriptors (HKS and WKS) for each keypoint.

3.2. Decomposition

Our decomposition algorithm relies on the distribution of keypoints on the shape's surface. Our algorithm performs a hierarchical clustering in the geodesic space of keypoints. More specifically, the algorithm first looks for large groups of keypoints, and recursively decomposes the groups into smaller groups. Each group of keypoints determines a mesh region. The decomposition ends when we cannot divide a group into smaller groups or when the area of the region that covers a group is very small relative to the area of the original shape. The outcome is a tree which represents the decomposition process. Fig. 1 shows a decomposition tree obtained with our algorithm. Furthermore, Algorithm 1 presents the pseudo-code of the decomposition algorithm in detail.

The input of the decomposition algorithm is a node T, which contains the original shape and all information computed in the pre-processing step (see Sec. 3.1). Technically, the output of this method is a tree with T as root node. The first part of the algorithm checks whether it is possible to continue decomposing a node (lines 3-13). The first check is through the comparison of areas between the shape at the input node and the original shape. If the mesh at node T is

Algorithm 1 GenerateTree(T)

Require: Node T
Ensure: Node T and its associated tree

```
1: areaOriginal ← mesh.getArea()
2: areaNode ← T.mesh.getArea()
3: if areaNode < 0.1× areaOriginal then
4:     isLeaf ← true
5: end if
6: if not isLeaf then
7:     Let R be the intra-cluster threshold.
8:     Let S be the inter-cluster threshold.
9:     C ← clustering(T.distancesKeypoints(), R, S)
10:    if |C| < 2 then
11:        isLeaf ← true
12:    end if
13: end if
14: if isLeaf then
15:    T.leaf ← true
16:    return T
17: end if
18: for each cluster c in C do
19:    Create a new node T_c
20:    (o_c, r_c) ← GetMin3DSphere(c.getKeypoints())
21:    k_c ← c.getMedoid()
22:    T_c.mesh ← T.mesh.getPatch(o_c, r_c, k_c)
23:    T_c.descriptor ← T.getPatchDescriptor(c.getKeypoints())
24:    Propagate keypoints from T to T_c
25:    Propagate keypoint distances from T to T_c
26:    Propagate keypoint descriptors from T to T_c
27:    T.children[c] ← T_c
28: end for
29: for each cluster c in C do
30:    for each cluster g in C do
31:        T.childrenDistance[c,g] ← d_g(k_c, k_g)
32:    end for
33: end for
34: for each cluster c in C do
35:    T.children[c] ← GenerateTree(T.children[c],δ)
36: end for
37: return T
```

too small, then T is marked as a leaf node. Subsequently, we propose to apply a medoid-based adaptive clustering to find groups of keypoints. These groups will generate regions with their associated new nodes. If it is not possible to obtain more than two clusters, the input node is marked as a leaf.

There are two aspects of our algorithm that deserve careful attention: the clustering algorithm (line 9) and the creation of new nodes (lines 18-28). We dedicate the following sections to describe these two points.

Adaptive Clustering with Medoids

Our algorithm is a variant of the adaptive clustering used to compute key-components in [15]. The main differences lie in two aspects. First, we use a medoid-based approach which prevents the computation of the multi-dimensional scaling. Second, we take an adaptive approach for determining the clustering thresholds depending on the hierarchical nature of our process.

Our method takes advantage of the distance matrix already computed in the pre-processing step and it is therefore not necessary to perform a multi-dimensional scaling to the keypoints. The clustering algorithm iterates over the set of keypoints assigning them to near clusters or creating new clusters otherwise. The decision of assigning a keypoint to a cluster or creating a new one depends on two parameters: R (intra-cluster threshold) and S (inter-cluster threshold).

These parameters depend on the area of the mesh in the input node, and are obtained using an empiric Gaussian function $R = 0.4 \times exp(-(areaRatio - 1)^2/0.5)$, where $areaRatio$ is the quotient between $areaNode$ and $areaOriginal$. This formulation was designed to control the clustering thresholds according to the size of the region. The Gaussian function distributes the values for R in the interval $[0.1, 0.4]$[1] depending on the area of the node region. This is consistent with the restriction in line 3. Additionally, the inter-cluster threshold S is always set to $2 \times R$. Therefore, the threshold values vary between $R = 0.4, S = 0.8$ and $R = 0.1, S = 0.2$.

Once the keypoints have been clustered, we need to compute the medoids of each resulting cluster. Let $C = \{c_1, \ldots, c_n\}$ be a cluster where each c_i is a keypoint, the medoid of C is defined as the keypoint that is approximately in the center of the distribution of the cluster. More formally,

$$medoid(C) = \arg\min_{c \in C} \sum_{k=1}^{n} d_g(c, c_k). \qquad (6)$$

In addition, if a cluster has a few elements (ten in our experiments), the cluster is removed. The algorithm repeats the assignment and update steps until reaching a number of iterations (for all our experiments, we use ten iterations). After the clustering, each cluster will generate a new node in the decomposition tree.

Node Creation

Several steps are performed to create a new node in the tree from a cluster c. First, the algorithm computes the smallest sphere which encloses the keypoints within the cluster. The

[1]The threshold values represent a fraction of the original area, so $R = 0.1$ really means $R = 0.1 \times areaOriginal$. We omitted this to facilitate the explanation.

obtained information is the sphere center o_c and a radius r_c. Second, we fetch the medoid of the cluster, i.e. the keypoint that is approximately at the center of the keypoint distribution of the cluster. Third, we extract a mesh patch using a growing region algorithm. This algorithm starts from the medoid keypoint and collects faces and vertices until covering the complete set of keypoints in the cluster. The region growing algorithm uses the radius r_c to determine the patch which certainly contains the keypoints. Fourth, we compute a descriptor for the new region. Let $C = \{c_1, \ldots, c_n\}$ be the set of keypoints of a cluster, each associated to a HKS descriptor. The descriptor of the region determined by the cluster C is

$$f_{region}(C) = \frac{\sum_{i=1}^{n} HKS(c_i)}{n}, \tag{7}$$

i.e. the average descriptor of the keypoint collection in the cluster. Fifth, all information about keypoints and geodesic distances are propagated from the parent node T to the new created node. Finally, the new node is stored as a child of the parent node T.

Subsequently, our decomposition algorithm computes the geodesic distances between regions (lines 29-33). We use the medoid keypoint as a reference to accomplish this goal. Let T_{c_1} and T_{c_2} two nodes resulting from the decomposition of the mesh at node T. The distance between the regions associated to T_{c_1} and T_{c_2} is

$$d_{reg}(T_{c_1}, T_{c_2}) = d_g(k_{c_1}, k_{c_2}), \tag{8}$$

where k_{c_1} and k_{c_2} stand for the medoids of clusters in nodes T_{c_1} and T_{c_2}, respectively. As a last step, our algorithm proceeds recursively for each child node (lines 34-36).

4. Hierarchical Matching

This section describes the algorithm to find the correspondences between two shapes. The algorithm 2 presents the pseudo-code of our method. This algorithm requires two trees T and P that represent the decomposition of two shapes as described in Sec. 3.2. The overall algorithm is based on the ability of matching regions in the internal nodes and keypoints in the leaf nodes.

The method is performed by depth levels (see Fig. 2). First, the matching of root nodes (level 0) implies finding correspondences between their children (level 1). Each correspondence generates a recursive call to the matching algorithm. The method proceeds until reaching the leaf nodes. In that case, we look for the best correspondence set between the keypoints in the leaf nodes. If at any time during the process, an internal node of a tree with a leaf node of the other is required to match, the internal node is treated as a leaf node. This is possible because every internal node contains the information to behave as a leaf node. Next, we describe how to perform the matching in the aforementioned cases.

Algorithm 2 Matching(T,P)

Require: Node T
Require: Node P
Ensure: A set of correspondences S
1: **if** not T.leaf and not P.leaf **then**
2: $Corr \leftarrow$ MatchingInternalNodes(T, P)
3: $S \leftarrow \{\}$
4: **for** each match $(t, p) \in Corr$ **do**
5: $L \leftarrow$ Matching(T.children[t], P.children[p])
6: $S \leftarrow S \bigcup L$
7: **end for**
8: **else**
9: $S \leftarrow$ MatchingLeafNodes(T, P)
10: **end if**
11: return S

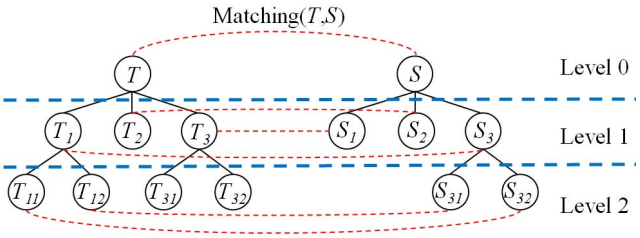

Figure 2. Representation of the matching process. The initial call to $Matching(T, S)$ (level 0) tries to find correspondences between the internal nodes in level 1. In the figure, the correspondences $\{(T_1, S_3), (T_2, S_2), (T_3, S_1)\}$ were found. Each pair generates a recursive call to the matching algorithm. The correspondence (T_1, S_3) causes a matching between internal nodes. In contrast, correspondences (T_2, S_2) and (T_3, S_1) means that algorithm will propagate the matching between internal nodes. Note that T_3 is not a leaf node. However it contains enough information to be considered as a leaf node, and therefore it can be matched to S_1.

4.1. Matching of Internal Nodes

Let T and S two internal nodes from different trees. Each node has children represented as $children(T) = \{t_1, \ldots, t_n\}$ and $children(S) = \{s_1, \ldots, s_m\}$, where T has n children and S has m children. Here we assume $n \leq m$ and therefore, the node with fewer children is matched to the other node. We define a boolean indicator variable as follows

$$x(i,j) = \begin{cases} 1, & \text{if } t_i \text{ matches } s_j \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Then, we formulate a quadratic optimization function as

follows:

$$F(x) = \alpha \sum_{i,j,i',j'} |d_{reg}(t_i, t_{i'}) - d_{reg}(s_j, s_{j'})| x(i,j)x(i',j') + \beta \sum_{i,j} \|f_{region}(t_i) - f_{region}(s_j)\|_2 x(i,j) + \gamma \sum_{i,j} |area(t_i) - area(s_j)| x(i,j) \quad (10)$$

where α, β and γ weight the contribution of each term in the function. The optimization function has two linear terms and a quadratic term. On the one hand, the linear terms evaluate the similarity between region descriptors and the consistency of areas. It is expected that two matched regions have similar descriptors and similar areas, indeed minimizing the linear terms. On the other hand, the quadratic term imposes a geometric consistency constraint. If there are two correspondences $x(i,j)$ and $x(i',j')$, it is expected that the geodesic distance between regions i and i' in one shape is quite similar to the geodesic distance between regions j and j' in the other shape. Finally, the goal is to obtain the minimizer of F

$$x* = \arg\min_x F(x), \quad (11)$$

subject to

$$\sum_i x(i,j) = 1 \ \forall j \text{ and } \sum_j x(i,j) \leq 1 \ \forall i. \quad (12)$$

The constraint in Eq. 12 controls the multiplicity of an element in the correspondence set. That is to say, every t_i only can correspond to a unique s_j, and vice versa.

4.2. Matching of Leaf Nodes

Unlike the matching of internal nodes, in the leaf nodes we need to find correspondences between keypoints. Let T and S two leaf nodes from different trees. Each node has a set of keypoints represented as $keypoints(T) = \{t_1, \ldots, t_n\}$ and $keypoints(S) = \{s_1, \ldots, s_m\}$, where T has n keypoints and S has m keypoints. Using Eq. 9 to define a boolean indicator variable, we formulate the optimization function for a leaf node as

$$L(x) = \alpha \sum_{i,j,i',j'} |d_g(t_i, t_{i'}) - d_g(s_j, s_{j'})| x(i,j)x(i',j') + \beta \sum_{i,j} \|WKS(t_i) - WKS(s_j)\|_2 x(i,j). \quad (13)$$

Note that the distances between regions in Eq. 10 have been replaced by geodesic distances between keypoints in Eq. 13. In addition, wave kernel signatures are used as descriptors for matching keypoints. Similar to the matching of internal nodes, our goal is to find a minimizer of L in the same way as Eq. 11 and the same constraints as Eq. 12.

5. Results and Discussion

We tested our method on the SHREC'2010 correspondence dataset [4]. The ground-truth of the correspondences was made available by authors for our evaluation. The dataset consists of three shapes (hereafter referred to as null shapes) and a set of transformed versions. The applied transformations are: isometry, holes, local scale, microholes, noise, sampling, scale, shotnoise, and topology. Each transformation is applied in five strength levels, leading to 45 transformed shapes for each null shape. For the quantitative results, we follow the methodology and notation proposed in [4]. We define our correspondence set as $C = \{(y_k, x_k)\}_{k=1}^M$, where M is the number of correspondences, and y_k and x_k are keypoints in the transformed and null shape, respectively. The ground-truth is composed by two correspondence sets C_0 and $\hat{C}_0$, containing the point-to-point correspondences for each vertex in a transformed shape and their symmetric counterpart, respectively.

The measure to quantify the quality of the correspondence set C is

$$D(C) = \frac{1}{M} \min \left\{ \sum_{k=1}^M d_g(x_k, x'_k), \sum_{k=1}^M d_g(x_k, x''_k) \right\} \quad (14)$$

where $(y_k, x_k) \in C$, $(y_k, x'_k) \in C_0$ and $(y_k, x''_k) \in \hat{C}_0$.

5.1. Experimental Setting

We detail our configuration as follows. First, we simplified the models to 10,000 vertices. The final correspondences were mapped back to the original shapes for evaluation. Second, we computed 100 Harris keypoints for each shape. Remember that this number can change after the filtering. Third, we use the original implementations of HKS[2] and WKS[3]. Fourth, for Eq. 10 we use the weights: $\alpha = 5 \times 10^{-4}$, $\beta = 1$, and $\gamma = 5 \times 10^{-2}$. Fifth, for Eq. 13 we use the weights: $\alpha = 5 \times 10^{-4}$, and $\beta = 1$. Finally, we used a branch-and-bound algorithm with LP-relaxation [2] to solve the integer quadratic programs.

5.2. Qualitative Results

We present results of our method in Fig. 3. The figures show examples with near-to-perfect localization of correspondences. An advantage of our method is the generation of very similar decomposition trees for near-isometric shapes. This enables the matching to be effective, yet fully exploiting the proposed hierarchical approach in favor of efficiency.

On the other hand, the addition of perturbations in meshes may affect the overall performance of finding correspondences. Figure 4 illustrates the correspondences found

[2]Available in http://www.geomtop.org/software/hks.html.

[3]Available in http://vision.in.tum.de/members/aubry/publications.

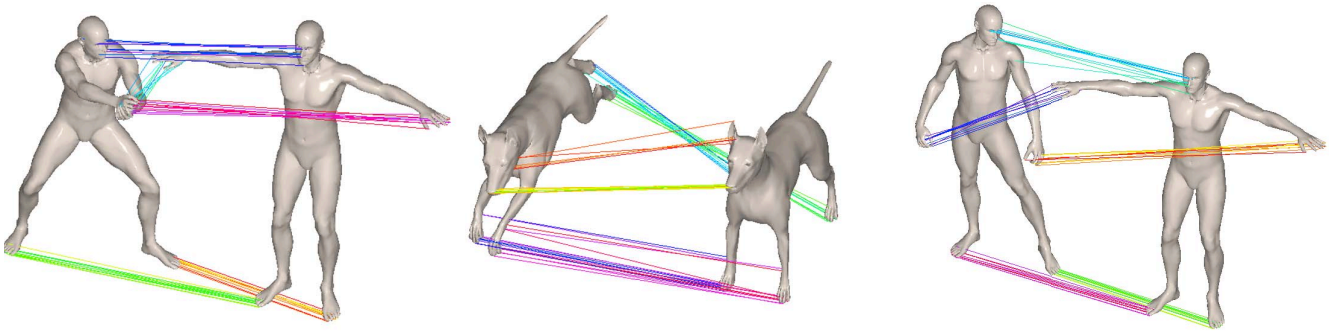

Figure 3. Correspondences found with our approach. Left: # correspondences = 66, geodesic error = 2.83, matching time = 0.12 sec. Middle: # correspondences = 58, geodesic error = 2.62, matching time = 0.08 sec. Right: # correspondences = 75, geodesic error = 3.51, matching time = 0.12 sec.

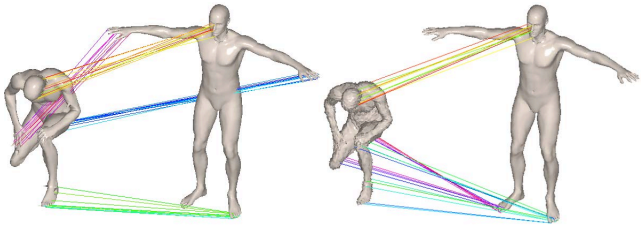

Figure 4. Correspondences found in presence of perturbations. Left: shotnoise, strength level 4 (# correspondences = 59, geodesic error = 5.23, matching time = 0.08 sec.). Right: noise, strength level 5 (# correspondences = 49, geodesic error = 5.42, matching time = 0.03 sec.).

with our algorithm in presence of strong transformations. Even in the presence of shotnoise (on the left) and Gaussian noise (on the right), our algorithm performs acceptably. Note that there are parts where correspondences were not found (for instance, the hands in the right figure). This is because perturbations affect the decomposition process, and particularly in this case, noise prevented the detection of robust regions of interest in hands. Nevertheless, other regions were well detected and interestingly those regions were well matched to the null shape. Therefore, the decomposition method delivers robust regions (and their associated keypoints) which arrive to reliable correspondences.

5.3. Quantitative Results

In this section, we present the performance of our method using the geodesic error described in Eq. 14 and compare it with state-of-the-art methods. We compare our method with the game-theoretic approach (the variant which merges correspondences gathered from 25 games, as reported in [11]) and the GMDS method as reported in the SHREC 2010 contest [4]. These two methods deliver on average 50 correspondences. Similarly, our method computes 45 correspondences on average.

Table 1 presents the average geodesic error of our method and the compared approaches with respect to the

	Strength				
Method.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
GMDS [5]	39.92	36.77	35.24	37.40	39.10
Game-theoretic [11]	10.28	12.51	11.73	14.35	18.26
Our method	**7.99**	**8.07**	**8.23**	**8.77**	**9.38**

Table 1. Average geodesic error of correspondences with respect to the strength level.

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	**9.37**	**7.28**	6.47	**6.11**	6.34
Topology	9.02	8.23	8.97	9.88	10.17
Holes	**7.13**	**6.46**	**6.45**	**7.55**	**8.50**
Micro holes	**5.86**	5.60	5.92	**6.05**	6.18
Scale	**6.45**	6.57	**7.30**	**7.60**	7.79
Local scale	9.45	**10.20**	**9.64**	**9.84**	**9.74**
Sampling	10.18	**11.62**	**12.96**	**15.78**	**19.20**
Noise	**5.65**	8.15	**7.79**	**8.02**	**8.58**
Shot noise	8.80	**8.50**	**8.56**	8.04	**7.88**
Average	**7.99**	**8.07**	**8.23**	**8.77**	**9.38**

Table 2. Average geodesic error per transformation and per level. Average number of correspondences: 45.

strength level of transformation. It is worth noting that our method significantly reduces the localization error for correspondences, especially in the stronger levels. For instance, our algorithm halves the error of the game-theoretic approach in the level 5.

Table 2 reports the average error per transformation and per strength level. Numbers in bold represent an improvement with respect to the state of the art. Our algorithm presents low error values in almost all transformations in strength levels 4 and 5. This is because the decomposition tree represents a mesh in an effective way, even in presence of severe perturbations. Hence this fact encourages us to think that our method is suitable for realistic applications. However, note that our method did not improve with respect to the topology transformation. The reason relies on the use

of geodesic distances for the geometric consistency, which are sensitive to topological changes. A solution would be the use of a more robust manner for measuring intrinsic distances (for instance diffusion distances). Also keep in mind that we have used the original versions of the heat and wave kernel signatures. Applying scale-invariant versions of these descriptors could further improve our results.

A note about execution time

Our matching algorithm takes on average 0.1 seconds to find the correspondences between two shapes. This means a speedup of 5x-40x with respect to the game-theoretic approach in [11]. All our experiments were run on a 64-bits Linux system with Intel Core-i7 (3.40GHz) processors and 32GB of RAM. Our algorithms were implemented in C/C++ with interfaces MEX/MATLAB.

6. Conclusions

We proposed a novel hierarchical approach to address the problem of finding reliable correspondences in non-rigid shapes. In our experiments, we showed that our method is robust to severe perturbations, making it suitable for realistic applications. Also, our approach outperformed the state of the art with respect to the localization error of correspondences. In addition, our matching algorithm is efficient, thanks to the use of the hierarchical structure of decomposition, which allows the reduction of the search space. In the future, we plan to use more robust descriptors and intrinsic distances to improve our results. Furthermore, a challenging direction is the adaptation of our algorithm to partial matching.

Acknowledgments

We thank Michael Bronstein for his extremely useful help with the SHREC benchmark. The work of Ivan Sipiran was partially supported by EC FP7 STREP Project PRESIOUS, Grant No. 600533. Also, this work was partially funded by Fondecyt (Chile) Project 1110111.

References

[1] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV Workshops*, pages 1626–1633, 2011.

[2] A. Bemporad, D. Mignone, and M. Morari. An Efficient Branch and Bound Algorithm for State Estimation and Control of Hybrid Systems. In *European Control Conference*, Aug. 1999.

[3] A. M. Bronstein. Spectral descriptors for deformable shapes. *CoRR*, abs/1110.5015, 2011.

[4] A. M. Bronstein, M. M. Bronstein, U. Castellani, A. Dubrovina, L. J. Guibas, R. P. Horaud, R. Kimmel, D. Knossow, E. von Lavante, D. Mateus, M. Ovsjanikov, and A. Sharma. SHREC'10: Robust correspondence benchmark. In *3DOR'10*, pages 87–91.

[5] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. *Proc. of the Natl. Acad. of Sci.*, pages 1168–1172, 2006.

[6] A. Dubrovina and R. Kimmel. Matching shapes by eigendecomposition of the Laplace-Beltrami operator. In *3DPVT*, 2010.

[7] R. Kimmel and J. A. Sethian. Computing Geodesic Paths on Manifolds. In *Proc. Natl. Acad. Sci. USA*, pages 8431–8435, 1998.

[8] Y. Lipman and T. Funkhouser. Möbius voting for surface correspondence. *ACM Trans. Graph.*, 28(3):72:1–72:12, July 2009.

[9] R. Litman, A. M. Bronstein, and M. M. Bronstein. Diffusion-geometric maximally stable component detection in deformable shapes. *Computers & Graphics*, 35(3):549–560, 2011.

[10] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. J. Guibas. One Point Isometric Matching with the Heat Kernel. *Comput. Graph. Forum*, 29(5):1555–1564, 2010.

[11] E. Rodolà, A. Bronstein, A. Albarelli, F. Bergamasco, and A. Torsello. A game-theoretic approach to deformable shape matching. In *CVPR*, pages 182–189, 2012.

[12] Y. Sahillioğlu and Y. Yemez. 3D Shape correspondence by isometry-driven greedy optimization. In *CVPR*, pages 453–458, 2010.

[13] A. Sharma, R. Horaud, J. Cech, and E. Boyer. Topologically-robust 3D shape matching based on diffusion geometry and seed growing. In *CVPR*, pages 2481–2488, 2011.

[14] I. Sipiran and B. Bustos. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, 27:963–976, 2011.

[15] I. Sipiran and B. Bustos. Key-components: detection of salient regions on 3D meshes. *The Visual Computer*, 2013. To appear.

[16] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *SGP'09*, pages 1383–1392. Eurographics Association, 2009.

[17] A. Tevs, A. Berner, M. Wand, I. Ihrke, and H.-P. Seidel. Intrinsic Shape Matching by Planned Landmark Sampling. *Computer Graphics Forum*, 30(2):543–552, 2011.

[18] A. Tevs, M. Bokeloh, M. Wand, A. Schilling, and H. P. Seidel. Isometric registration of ambiguous and partial data. In *CVPR*, pages 1185–1192, 2009.

[19] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. Surface feature detection and description with applications to mesh matching. In *CVPR*, pages 373–380, 2009.

[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 *CVPR*, pages 382–389, 2010.

[21] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. In *SGP'08*, pages 1431–1439. Eurographics Association, 2008.