Graphical Models 97 (2018) 17–29

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

An evaluation of canonical forms for non-rigid 3D shape retrieval

David Pickup*[a], Juncheng Liu[a,b], Xianfang Sun[a], Paul L. Rosin[a], Ralph R. Martin[a], Zhiquan Cheng[c], Zhouhui Lian[b], Sipin Nie[d], Longcun Jin[d], Gil Shamai[e], Yusuf Sahillioğlu[f], Ladislav Kavan[g]

[a] Cardiff University, UK
[b] Peking University, China
[c] Avatar Science (Hunan) Company, China
[d] South China University of Technology, China
[e] Technion - Israel Institute of Technology, Israel
[f] Middle East Technical University, Turkey
[g] University of Utah, US

ARTICLE INFO

Keywords:
Geometry processing
Canonical forms
3D Shape retrieval

ABSTRACT

Canonical forms attempt to factor out a non-rigid shape's pose, giving a pose-neutral shape. This opens up the possibility of using methods originally designed for rigid shape retrieval for the task of non-rigid shape retrieval. We extend our recent benchmark for testing canonical form algorithms. Our new benchmark is used to evaluate a greater number of state-of-the-art canonical forms, on five recent non-rigid retrieval datasets, within two different retrieval frameworks. A total of fifteen different canonical form methods are compared. We find that the difference in retrieval accuracy between different canonical form methods is small, but varies significantly across different datasets. We also find that efficiency is the main difference between the methods.

1. Introduction

The ability to recognise a deformable object's shape, regardless of the pose of the object, is an important requirement for modern shape retrieval methods. One approach to this problem is to transform each deforming model into a canonical form which (ideally) factors out the pose, leaving a standard pose-independent version of the shape. This allows rigid shape retrieval algorithms to be used for non-rigid shape retrieval. Many different methods have been proposed for automatically computing a canonical form from a 3D mesh. Methods using such approaches along with rigid retrieval systems have performed well on shape retrieval benchmarks [21]. However, most of these methods have not been compared using the same dataset, or used for retrieval within the same rigid retrieval system, so their relative performance is unclear. We recently proposed a new benchmark to provide a meaningful comparison of existing and new canonical form methods for non-rigid shape retrieval [34]. In this paper we extend our previous work to test a larger number of recent canonical form methods, allowing us to more reliably characterize the state-of-the-art performance. Our previous benchmark used a small dataset. In this work we test the performance of all canonical form methods on three larger recent non-rigid object datasets. The algorithms were tested by using their output canonical forms within two different rigid shape retrieval frameworks, and evaluating the retrieval performance.

Our paper is structured as follows. Section 2 describes the datasets used. Section 3 outlines all the canonical form methods tested. Section 4 outlines the retrieval frameworks used in our experiments. Our experiments are presented in Section 5, and finally we conclude in Section 6.

2. Datasets

We performed experiments on five different datasets. Two of these datasets contain only non-rigid human models (see Sections 2.1 and 2.2), and the third contains a variety of different non-rigid shapes (see Section 2.3). We also have non-rigid shape dataset TOSCA and for robustness test we make the noisy version of TOSCA.

2.1. Real human dataset

The *Real* human dataset [30,31] was built from point-clouds contained within the Civilian American and European Surface Anthropometry Resource (CAESAR) [7]. Meshes were fitted to these point-clouds, and a data-driven technique was used to create a set of poses for

* Corresponding author.
E-mail addresses: david.pickup@bath.edu, david@bristolalumni.org.uk (D. Pickup).

https://doi.org/10.1016/j.gmod.2018.02.002
Received 7 April 2017; Received in revised form 5 February 2018; Accepted 15 February 2018
Available online 07 March 2018
1524-0703/ © 2018 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/BY/4.0/).

each human subject. The *Test* set contains 400 models, representing 40 human subjects (20 male, 20 female), each in 10 different poses. The *Training* set contains 100 models, representing 10 human subjects (5 male, 5 female), again in 10 different poses. None of the training subjects or poses are present in the test set. The meshes each have approximately 15,000 vertices, varying slightly from mesh to mesh. A selection of these models is shown in Fig. 1(a). This dataset is freely available to download [29].

2.2. Synthetic human dataset

The *Synthetic* human dataset [30,31] was created using a parameterized human model, where parameters control body shape, provided as part of the DAZ Studio [8] 3D modelling and animation software. Different poses were created using a palette of poses also provided with DAZ Studio. The *Test* set contains 300 models, representing 15 different human subjects (5 male, 5 female, 5 child), each in 20 different poses. The *Training* set contains 45 models, representing 9 human subjects (3 male, 3 female, 3 child) in 5 different poses. None of the training subjects or poses is present in the test set. The meshes have approximately 60,000 vertices, again varying slightly. A selection of these models is shown in Fig. 1(b). This dataset is also freely available to download [29].

2.3. Non-rigid shapes dataset

The non-rigid shapes dataset [22] contains a greater variety of shape classes, and therefore we also performed experiments on this dataset. It contains 1200 meshes, split into 50 different shape classes. Each shape class contains 24 meshes of an object in different non-rigid poses. Four meshes in each shape class contain topological errors, such as disconnected components, or unwanted connections. The average number of vertices per mesh is 9,607. A selection of these models is shown in Fig. 1(c). This dataset is also freely available to download.[1]

2.4. TOSCA and TOSCA with noise

TOSCA [5] contains hi-resolution three-dimensional nonrigid shapes in a variety of poses for non-rigid shape similarity and correspondence experiments. The database contains a total of 80 objects, including 11 cats, 9 dogs, 3 wolves, 8 horses, 6 centaurs, 4 gorillas, 12 female figures, and two different male figures, containing 7 and 20 poses.

We also built a noisy version of TOSCA by adding random turbulence to vertices of the mesh. This is to test the robustness of the methods.

3. Canonical form methods

We now briefly describe each of the canonical form methods compared in our study. Many are variants of the multidimensional scaling (MDS) and geodesic distance based method proposed by Elad and Kimmel [11], but some replace either or both of these components.

3.1. Background and the classical multidimensional scaling

The key observation is that the pairwise geodesic distances remain the same for the same shape with two different poses (known as isometric deformation). Transforming the geodesic distances to an Euclidean space could eliminate the differences of pose. The classical MDS method takes as input the pairwise distances, and produces a configuration (or coordinates) of all the vertices that best preserves the provided distances.

[1] http://www.icst.pku.edu.cn/zlian/shrec15-non-rigid/data.html.

Elad and Kimmel [11] proposed computing a canonical form of a mesh by mapping the geodesic distance between all pairs of vertices to three-dimensional Euclidean distances. As the geodesic distances are pose invariant, the Euclidean embedding is a pose invariant representation of the shape.

We first compute the all-pairs geodesic distance matrix of the mesh using the fast marching method [18]. The Euclidean embedding of these distances is then computed using classical MDS. We do this by first calculating

$$B = -\frac{1}{2}JDJ, \tag{1}$$

where

$$J = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T, \tag{2}$$

$$\mathbf{1}_{N\times 1} = [1,\ 1, \cdots, 1]^T, \tag{3}$$

D is the matrix of pairwise squared geodesic distances, and N is the number of mesh vertices. I is an $N \times N$ identity matrix, J is a centralization operator and B is the centralized squared geodesic distance matrix. We then compute the four largest eigenvalues ($\lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \lambda_3$) and the associated eigenvectors (ϕ_0, ϕ_1, ϕ_2, ϕ_3) of B. Given a point $\mathbf{p}$ on the mesh, the MDS embedding is calculated as

$$\text{MDS}(\mathbf{p}) = \left(\lambda_1^{\frac{1}{2}}\phi_1(\mathbf{p}),\ \lambda_2^{\frac{1}{2}}\phi_2(\mathbf{p}),\ \lambda_3^{\frac{1}{2}}\phi_3(\mathbf{p})\right). \tag{4}$$

Given the geodesic distances, the MDS method has a complexity of $O(N^2)$, where N is the number of vertices. However, the computation of geodesic distances has a time complexity of $O(N^2 \log N)$. Due to this high computational expense the meshes are simplified to approximately 2000 vertices before computing the canonical forms. The methods proposed in Sections 3.2–3.4 also need to compute geodesic distances, and likewise use simplified meshes.

3.2. Fast multidimensional scaling

The fast MDS canonical form method [13] uses a variant of the method in Section 3.1, in which the geodesic distances are projected into Euclidean space one dimension at a time. We first compute the geodesic distance between each pair of vertices i and j as d_{ij}. For each of the three dimensions in turn we select the two most distant vertices O_a and O_b (in Euclidean distance). All other vertices are then projected onto the line O_aO_b:

$$x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{bi}^2}{2d_{ab}}. \tag{5}$$

x_i is used as the embedded vertex coordinates in the current dimension. The distances are then updated as

$$d_{ij}^{2'} = d_{ij}^2 - \|x_i - x_j\|^2. \tag{6}$$

This MDS method has a lower time complexity of $O(N)$, but overall this approach still requires the expensive $O(N^2 \log N)$ computation of geodesic distances.

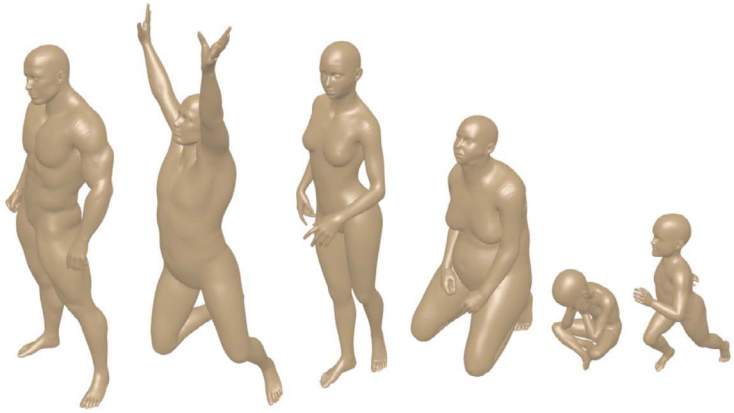
3.3. Least squares multidimensional scaling

This is another variant on the method described in Section 3.1, also proposed by Elad and Kimmel [11]. We use the *SMACOF* (scaling by majorizing a convex function) algorithm to compute the MDS. SMACOF minimises the following functional:

$$S(X) = \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}(\delta_{ij} - d_{ij}(X))^2, \tag{7}$$

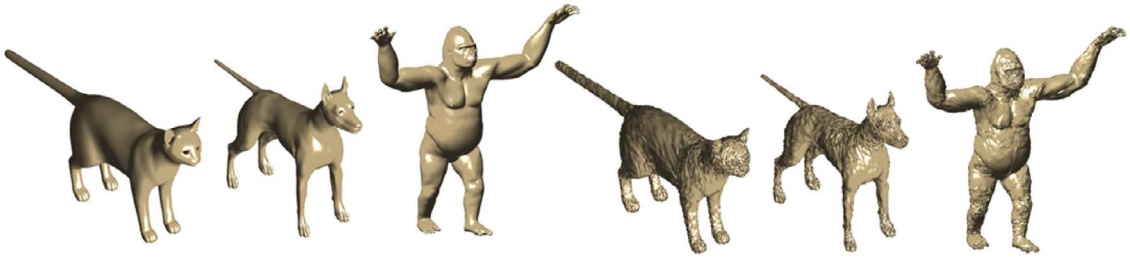
where N is the number of vertices, w_{ij} are weighting coefficients, δ_{ij} is

(a) Real human dataset.



(b) Synthetic human dataset.

(c) Non-rigid shapes dataset.



(d) tosca shapes dataset.

Fig. 1. A selection of models from each dataset.

the geodesic distance between vertices i and j of the original mesh, and d_{ij} is the Euclidean distance between vertices i and j of the resulting canonical mesh X. The stress function is minimised iteratively using the code provided by Bronstein et al. [4].

This MDS algorithm has a complexity of $O(N^2)$ and also depends linearly on the number of iterations, but again all pairs of geodesic distances must be found.

3.4. Non-metric multidimensional scaling

This method is very similar to the one in Section 3.3, but instead of matching the Euclidean distances to the exact geodesic distances, we only match the ordering of the distances. The stress function we minimise for this method is

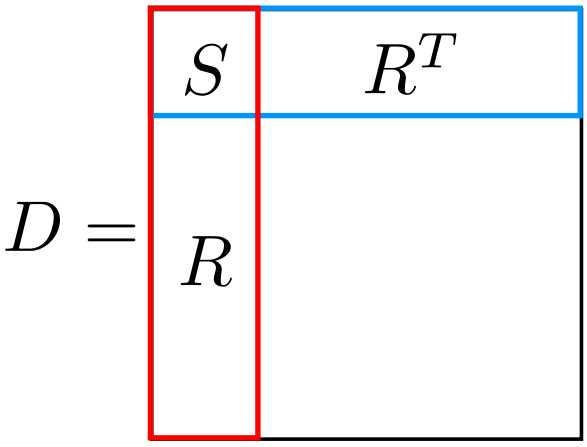


Fig. 2. Partition of D.

$$S(X) = \frac{\sum_{i=1}^{N}\sum_{j=i+1}^{N}(f(\delta_{ij}) - d_{ij}(X))^2}{\sum_{i=1}^{N}\sum_{j=i+1}^{N} d_{ij}(X)^2}, \tag{8}$$

where N is the number of vertices, δ_{ij} is the geodesic distance between vertices i and j of the original mesh, d_{ij} is the Euclidean distance between vertices i and j of the resulting canonical mesh X, and f is an optimal monotonic function of the dissimilarities. This MDS method is less restrictive than metric MDS, and some researchers have found that it produces more desirable results [17], but again requires geodesic distances.

3.5. Accelerated MDS

Shamai et al. [40] proposed an acceleration framework for MDS, which accurately approximates the pairwise geodesic distance maps, which reduces the high complexities to quasi-linear.

Given a triangular mesh representing a 3D object, let $Z_{p\times k}$ be the matrix holding the points $\{z_i\}_{i=1}^{p}$ of the canonical form in its rows. Classical scaling finds the canonical form Z by minimization:

$$\arg_Z \min \left\| ZZ^T + \frac{1}{2}JDJ \right\|_F,$$

where D_{ij} holds the squared geodesic distance between points i and j in the mesh. The main drawback of classic MDS is its use of the matrix D, which is too large to be computed and stored. In this method, termed NMDS [40], the Nyström method [43] is improved by adding a regularization term, and using it to approximate the matrix D, by decomposing it into a product of smaller matrices. For this method, only n columns of the matrix D need to be computed (typically 10–20 columns), corresponding to n landmarks chosen from the mesh. Each column corresponds to the squared geodesic distances between a particular landmark and the rest of the mesh points. The landmarks are chosen using *farthest point sampling* [15] and the geodesic distances are computed using the *fast marching method* [18]. The *NMDS* method is summarized in Procedure 1.

3.6. Constrained MDS

Sahillioğlu [38] presented a method that introduces a constraint on the original MDS formulation in order to preserve the initial geometric details in the output shape. The motivation for preserving details during the MDS embedding process is to better distinguish semantically similar objects with varying geometric details, which is potentially useful in a shape retrieval application.

Specifically, this method exploits the perfectly accurate bijection between the original shape and its Landmark MDS embedding [9], where the former and the latter has no and significant geometric

Input $G = \{D, V\}$ with $p = |V|$ vertices, parameters n, k.
Output A matrix Z which contains the coordinates of the embedding into $\mathbb{R}^k$.
1: Choose n samples of the data using *farthest point sampling* and obtain the matrix F.
2: Compute R, by element-wise squaring F, $R_{ij} = F_{ij}^2$.
3: Denote by S the intersection of R and R^T in D (see Figure 2).
4: Compute $\tilde{V}$ and $\tilde{\Lambda}$, which contain the $n_1 = \text{round}(n/2)$ largest eigenvalues and corresponding eigenvectors of S, using eigenvalue decomposition
5: Compute $QW = J\tilde{R}\tilde{V}$ using QR factorization
6: Compute $\tilde{V}_2$ and $\tilde{\Lambda}_2$, which contain the k largest eigenvalues and corresponding eigenvectors of $-\frac{1}{2}W\tilde{\Lambda}^{-1}W^T$, using eigenvalue decomposition.
7: Return the coordinate matrix $Z = Q\tilde{V}_2\tilde{\Lambda}_2^{\frac{1}{2}}$

Algorithm 1. Nyström-MDS.

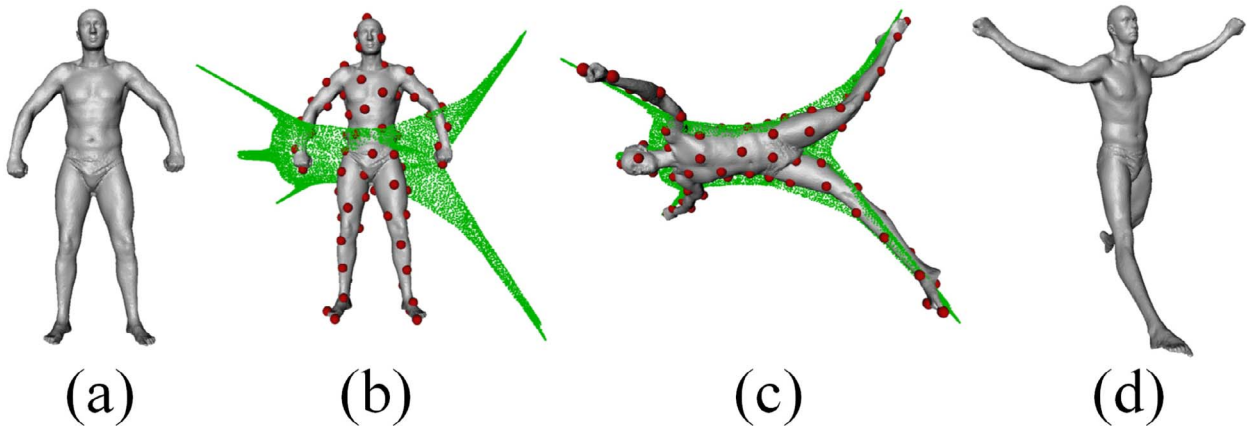


Fig. 3. (a) Original shape. (b-c) Classical MDS [12] on landmarks (spheres) is interpolated via Landmark MDS [9] (green, distorted). (d) Output shape produced by Sahillioğlu [38]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

distortions, respectively. By moving the original vertices towards the corresponding embedding vertices in the presence of a deformation regularization energy, it is possible to obtain a detail-preserving MDS pose as depicted in Fig. 3. While the method is accurate and fast, it requires a tetrahedralization preprocessing step [16] and relies on geodesic distances that alter drastically under topological noise, e.g., hands connected by noisy edges. The execution of this algorithm on a 3.40GHz PC with 8GB RAM is about 22 s for a mesh with 45K vertices, preceded by 44 s of tetrahedralization. The usage of the problematic geodesic distances is avoided in a recent work [39], which in turn alleviates the sensitivity to topological noise.

3.7. Detail-preserving mesh unfolding (Yusuf's method)

The constrained MDS [37] obtains a detail-preserving canonical form by treating the vertices of the Landmark MDS embedding [10] as handles to deform the original shape. However, their work employs the problematic geodesic distances and a simpler yet faster deformation regularization energy. Instead, they proposed a finite elements based method [39] which does not use geodesics and achieves regularization with a more sophisticated scheme based on springs and the finite element method, which in turn achieves more accurate results in terms of element inversions and retrieval performance.

Unlike classical approaches, such as least-squares multidimensional scaling, the method preserves the geometric details of the input shape in the resulting shape. The optimization framework, fed with a triangular or a tetrahedral mesh in 3D, tries to move each vertex as far away from each other as possible subject to finite element regularization constraints. Intuitively this effort minimizes the bending over the shape while preserving the details. Avoiding geodesic distances in computation renders the method robust to topological noise.

3.8. Global point signatures

We have implemented the method to compute the Global Point Signatures (GPS) embedding of mesh proposed by Rustamov [36]. Firstly we calculate the discrete Laplace–Beltrami operator with cotangent weights [14] on the mesh. We then compute the four smallest eigenvalues ($\lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$) and their associated eigenvectors (ϕ_0, ϕ_1, ϕ_2, ϕ_3) of the Laplace-Beltrami operator. Given a point $\mathbf{p}$ on the mesh, the GPS embedding is calculated as a combination of the first three non-constant eigenvectors ϕ_{1-3}:

$$\mathrm{GPS}(\mathbf{p}) = \left(\frac{1}{\sqrt{\lambda_1}}\phi_1(\mathbf{p}), \frac{1}{\sqrt{\lambda_2}}\phi_2(\mathbf{p}), \frac{1}{\sqrt{\lambda_3}}\phi_3(\mathbf{p}) \right). \tag{9}$$

The eigenspaces of the Laplace–Beltrami operator are invariant to metric preserving deformations, and therefore the GPS embedding is a pose invariant representation of the mesh. This method is fast to compute, as it avoids the computation of geodesic distances.

3.9. Skeleton driven canonical forms

A variant on the canonical forms presented by Elad and Kimmel [11] is used [33]. A canonical form is produced by extracting a curve skeleton from a mesh, using the method by Au et al. [1] (Fig. 4(a)). The SMACOF Multidimensional Scaling method used by Elad and Kimmel [11] is then applied to the skeleton, to put the skeleton into a canonical pose (Fig. 4(b)). The skeleton driven shape deformation method by Yan et al. [44] is then used to deform the mesh to the new pose defined by the canonical skeleton (Fig. 4(c)). This produces a similar canonical form to Elad and Kimmel [11], but with the local features better preserved.

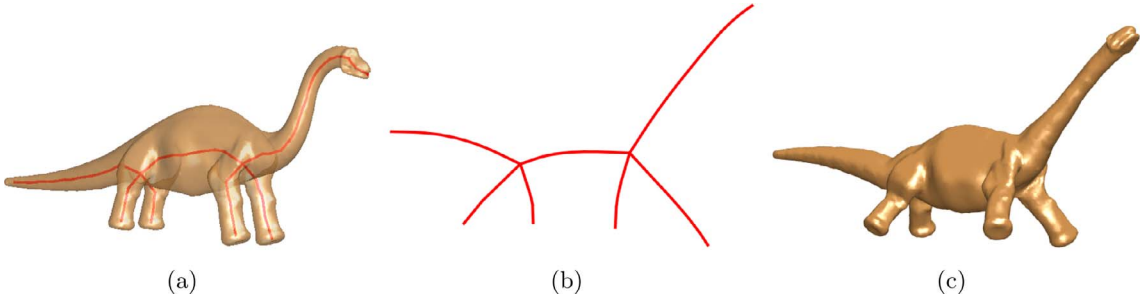


Fig. 4. Outline of the skeleton-based method. (a) Extract the shape's skeleton. (b) Compute the canonical skeleton. (c) Deform the mesh using the skeleton.

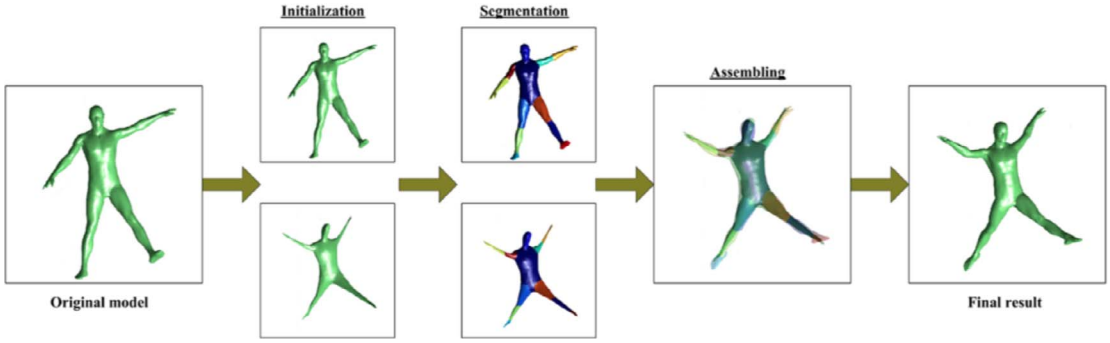


Fig. 5. An overview of Lian's method [24].

3.10. Euclidean distance based canonical forms

The Euclidean distance based method by Pickup et al. [32] computes a canonical form of a mesh by stretching out its limbs so that its extremities are distant from one another. This is achieved more efficiently than using multidimensional scaling (MDS) on the geodesic distances [11]. It instead maximises the Euclidean distances between feature points on the extremities of the mesh, whilst preserving the original edge lengths to ensure isometric deformation.

We first scale the mesh so that the maximum distance of any point on its surface to the centroid of all vertices is one. We then use the method of Ben-Chen and Gotsman [3] to calculate the conformal factor of the mesh. The conformal factor increases along the length of mesh protrusions, resulting in high values at mesh extremities. To obtain a set of feature points for a mesh with N vertices, we sample the $\sqrt{N}$ vertices which have the largest conformal factors and also satisfy the requirement that they are local maxima (their conformal factors are greater than that of all neighbours in a 2-ring neighbourhood). Having $\sqrt{N}$ feature points means we are able to compute the Euclidean distances between all pairs of feature points in $O(N)$ time.

We compute the canonical form of the mesh by adapting the least-squares MDS formulation used by Elad and Kimmel [11] Eq. (7), so that the value of δ_{ij} for all connected vertices i and j is set equal to the length of the edge connecting them. This aims to preserve the edge lengths of the mesh, to ensure isometric deformation. In order to maximise the distance between feature points, the value of δ_{ij} for each pair of the $\sqrt{N}$ sampled vertices is set to 10, as suggested by Pickup et al. [32]. As long as this value is large enough and the parameter β, discussed below, is optimised accordingly, any value can be chosen. If two vertices i and j are neither a pair of feature points nor connected by an edge, we do not enforce a target distance between them, so w_{ij} is set to zero. Not having to compute and optimise the distances between these points is crucial in keeping the linear time complexity of the distance calculations.

The weights w_{ij} in Eq. (7) for all i and j that are connected by an edge are set to β/δ_{ij}^2, where β is a user defined parameter for preserving edge lengths. We divide by the square of the edge length δ_{ij}^2 so that the distance in Eq. (7) becomes a relative, rather than absolute, difference, making the weighting independent of the length of the edge. The conformal factor is normalised to lie in the interval [0, 1], and the entries in the weighting matrix w_{ij} for each pair of feature points are set to the mean of their conformal factors. This results in vertices which are nearer the ends of long 'limbs' of the object having a higher impact on the resulting canonical form, and avoids stretching out inappropriate parts of the mesh. The SMACOF algorithm can then be used to minimise Eq. (7) as described in [11].

In many cases there are fewer local maxima of the conformal factor than $\sqrt{N}$. We want the number of feature points to be exactly $\sqrt{N}$ so that the number of edges connecting pairs of feature points grows at the same rate as the number of mesh vertices. This in turn ensures that we can use the same value for the parameter β regardless of mesh resolution. Pickup et al. [32] offer two different solutions to handling this issue. The first is to increase the number of feature points to $\sqrt{N}$ by adding extra randomly selected vertices as feature points. We refer to this solution as "Euclidean Random" in Section 5.

The second is to separately normalise the weightings w_{ij} used for pairs of feature points, and for adjacent vertices. We normalise the weights for adjacent vertices by dividing by the total number of edges, and we normalise the feature point pair weights by dividing by the sum of all feature point pair weights. We refer to this solution as "Euclidean Normalised" in Section 5.

This method eliminates the need for expensive geodesic distance computation. Feature point selection and Euclidean distance calculations have time complexity of $O(N)$, and the distance computations for each iteration of the MDS stress minimisation algorithm also take $O(N)$ time.

3.11. Feature-preserving canonical form (Lian's method)

Lian et al. [24] proposed a MDS-based method that yet much better preserves the detail features compared to the original algorithm. Their method first constructs the standard MDS canonical forms for each model. Then the models are segmented into individual components using random walk [20]. After the segmentations, each component is moved according to the guidance of the MDS canonical forms and finally these components are connected together by an as-rigid-as-possible method. The method uses the standard MDS canonical forms as guidance while preventing the distortions by component-wise movements. The performance of the method, however, largely depends on the segmentation's accuracy. An overview of the method is illustrated in Fig. 5.

3.12. Mesh unfolding using semidefinite programming (Liu's method)

Liu et al. [26] proposed an automatic mesh unfolding method by optimizing an objective function using semidefinite programming. Motivated by the Maximum Variance Unfolding (MVU) [42] algorithm, the method formulates the total vertex variance to the trace of the Gram matrix. Furthermore, the method integrates a locally linear preserving term into the objective function, which leads to an enhancement of the detail-preserving ability.

The basic idea is to maximize the total variance of the vertex set for a given 3D mesh, while preserving the details by minimizing locally linear reconstruction errors. By optimizing a specifically-designed objective function, the vertices tend to move against each other as far as possible, which leads to the unfolding operation. Compared to other Multi-Dimensional Scaling (MDS) based unfolding approaches, their method achieves a stronger degree of unfolding and does not require geodesic distance calculation.

The overall variance of the vertex set can be formulated using the trace of the Gram matrix as:

$$\begin{aligned} \underset{\mathbf{K}}{minimize} \quad & tr(\mathbf{K}(\mathbf{G}-\alpha\mathbf{I})) \\ s.\,t. \quad & \sum_{ij}\mathbf{K}_{ij}=0 \\ & \mathbf{K}\in\mathscr{S}_n^+ \\ & \forall \ <\boldsymbol{v}_k,\boldsymbol{v}_j>\in\mathscr{E}, \\ & \mathbf{K}_{k,k}+\mathbf{K}_{j,j}-2\mathbf{K}_{k,j}=\mathbf{S}_{kj}. \end{aligned} \tag{10}$$

where $\mathbf{K}$ is the Gram matrix and hence belongs to the semidefinite matrix set. The length of each edge on the mesh is preserved by adding the constraints $\mathbf{K}_{k,k}+\mathbf{K}_{j,j}-2\mathbf{K}_{k,j}=\mathbf{S}_{kj}$ where $\mathbf{S}_{k,j}$ is the squared length of edge $<\boldsymbol{v}_k,\boldsymbol{v}_j>$. The term $\mathbf{G}$ is the locality linear preserving term and the parameter α balances the unfolding term and detail-preserving term.

4. Retrieval frameworks

We use two local feature based frameworks from the existing literature to produce retrieval results, one based on extracting features from rendered depth maps of the shape (see Section 4.1), and the other based on extracting features from the 3D mesh itself (see Section 4.2).

4.1. Clock matching and bag-of-features

The Clock Matching and Bag-of-Features (CM-BOF) framework was proposed by Lian et al. [23], and has been shown to perform extremely well at non-rigid 3D shape retrieval when combined with a canonical form method [21]. To compute a descriptor of a 3D shape we first centre the mesh, normalise its scale, and use a combination of principal component analysis (PCA) and rectilinearity [25] to normalise its orientation. We then render 66 depth images of the mesh from viewpoints at the vertices of a soccer-ball. SIFT features [27] are then extracted from the depth images. A 1000 length histogram descriptor of each image of the shape is then created using the bag-of-words method. The similarity of two shapes is calculated as the sum of the similarities of their matching views.

4.2. 3D Spatial pyramids

We also use the 3D Spatial Pyramids (3DSP) framework proposed by Redondo–Cabrera et al. [35]. To describe a shape, we first extract a set of 3D SURF features [19] from the 3D mesh itself. The volume containing the mesh is recursively broken down into sub-cubes, forming a spatial pyramid. A bag-of-words is then computed for each part of the pyramid and then the histograms are concatenated to form the shape's final descriptor. The similarity of two shapes is computed as the similarity of their descriptors.

The reason why we adopt this feature is that it is a shape-based descriptor. Namely, it directly extracts descriptors on 3D mesh while CM-BOF deals with the rendered views of 3D mesh and therefore a view-based feature. We use both types of feature to test the adaptability of different canonical forms.

5. Experiments

Here we detail the results of our experiments. We show a visual evaluation of the canonical forms in Section 5.1, the retrieval task used to evaluate the canonical forms is described in Section 5.2, the retrieval performance is presented in Section 5.3, and timings for each method are given in Section 5.4.

5.1. Visual evaluation

Two example canonical forms produced by each method for each dataset are shown in Figs. 6–8. All canonical forms share the property that the limbs of the objects are straightened out. The different methods achieve this with varying degrees of success; there appears to be a trade-off between a fully straightened out pose, and preserving detailed shape features. Most MDS-based methods, and the GPS method, successfully straighten out the limbs of the models, but cause a large amount of distortion to the shape. The Constrained MDS and Skeletons methods are the two exceptions to this: while they both rely on MDS, they manage to preserve much of the shape detail. The GPS method causes so much distortion, it is extremely difficult, if not impossible, to visually recognize the shape of the object from its canonical form. Yusuf's method constructs canonical forms by solving a finite elements problem which requires the tetrahedralization of mesh. Their method significantly preserves the details. However, when handling non-watertight mesh, preprocessing is needed. Liu's method yields canonical forms by maximizing the vertex variance while preserving the edge length. Their method exhibits distortions at limb ends.

Some methods cause various other artifacts. The Constrained MDS method fails to completely transform some of the models, such as the man lying on his side. This method also exhibits curves and bends in the limbs of the objects. The Euclidean Random and Normalised methods do not produce as consistent a pose across different meshes, and also cause some "spiky" parts in the objects, where feature points do not lie at the mesh extremities. The Skeletons method can result in bulges at joints which were highly bent in the original mesh, and the faces of the *Synthetic* humans are highly distorted due to the complex skeleton formed in these areas because of the detailed geometry of the eyes, mouth and tongue.

Some of the meshes in the *Non-Rigid* dataset contain topological errors. Figs. 9 and 10 show three examples of these, where parts of the meshes have been incorrectly fused together, and the resulting canonical forms. All the canonical forms fail to cope with such topological errors. The method by Boscaini et al. [6] show they can handle some small topological errors, but we were unable to test their method as there is no publicly available implementation.

5.2. Retrieval task and evaluation method

The canonical forms for each method were used in turn for shape retrieval. The retrieval task is defined as:

> Given a query model, return a list of all models, ordered by decreasing shape similarity to the query.

Every model in the database was used in turn as a separate query model.

The evaluation procedure used to assess the results (see Section 5.3) is similar to that used by previous comparative studies [21,22]. We evaluate the results using various statistical measures: nearest neighbour (NN), first tier (FT), second tier (ST), discounted cumulative gain (DCG), and precision and recall curves. Definitions of these measures are given in [2,41].

5.3. Retrieval results

Tables 1–5 show the retrieval performance for both retrieval frameworks applied to each set of canonical forms. In general the retrieval results were higher when using the view-based CM-BOF retrieval framework. This shows that view-based frameworks may be more suited for use with canonical forms than frameworks based on local 3D features.

All the canonical form methods achieved a significantly higher retrieval performance for the *Non-Rigid* datasets, performing poorly on both the *Real* and *Synthetic* human datasets. The human datasets were designed to be more challenging for retrieval than the more general *Non-Rigid* dataset. In general other methods not based on canonical forms achieve a much better result than ones based on canonical forms for such data [31]. As might be expected, for all datasets, using

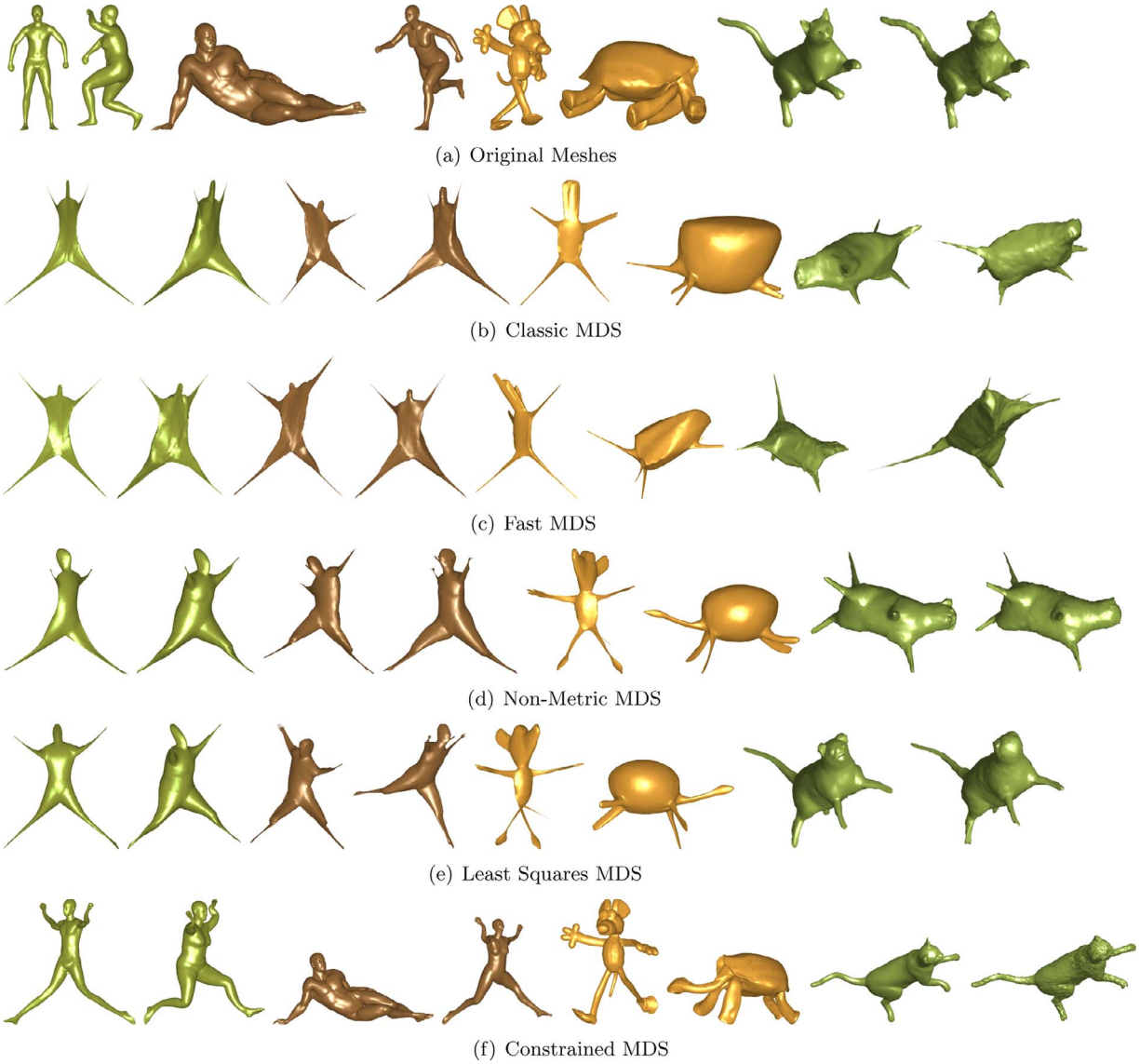


Fig. 6. Two example canonical forms for several methods from Real, Synthetic, Non-rigid along with two examples from TOSCA and TOSCA with noise respectively.

canonical forms achieved a better result than simply running rigid retrieval on the original meshes, which supports the claim that canonical forms improve results by adding some pose-invariance.

The results on the *Non-Rigid* dataset show very high retrieval performance, with the best methods achieving close to perfection. There is a general issue with the low difficulty of this dataset, which is highlighted by the good results also achieved by performing retrieval directly on the original meshes, with no attempt at pose-invariance. This shows that the difference in shape between classes is likely larger than the difference due to within class pose variation, and indicates the rather unsuitable nature of this benchmark.

The GPS method achieved the best retrieval performance across all measures for the *Real* dataset, and the highest NN and DCG scores for the *Synthetic* dataset. The difference in NN performance for this method over the others on the *Synthetic* dataset is highly significant, with GPS achieving 0.4 for CM-BOF retrieval, whilst the next best score being only 0.14. This is surprising, as the GPS method appears to produce the largest distortions in the shapes, and achieves a significantly lower performance than all other methods on the *Non-Rigid* dataset. In fact, the GPS method is the only method to perform worse than performing retrieval directly with the original meshes on the *Non-Rigid* dataset.

The Least Squares MDS method achieved the best performance across all measures for the *Non-Rigid* dataset. The Non-metric MDS, Constrained MDS, and Skeletons methods all achieve the same NN performance, and only a slightly lower performance across the other measures.

We also evaluate the methods by using the TOSCA dataset and its noisy version. The noisy dataset is obtained by adding vertex-wise uniformly distributed random turbulence in an amplitude much smaller than the shape diameters (1% of the shape diameter). We conduct this experiment to test the robustness of each methods. The performance of TOSCA and TOSCA with noise are collected in Tables 4 and 5 respectively. As expected the best performance in every measure suffers from an insignificant drop due to the added noise. And the methods appear to have non-uniform reactions to noise. On TOSCA, the Constrained MDS achieves the best results while on the noisy TOSCA it drops down. Methods such as Lian's feature-preserving canonical forms, Yusuf's detail-preserving mesh unfolding and CMDS are less sensitive to noise. That Yusuf's method survives on topological noise is due to the elimination of geodesics.

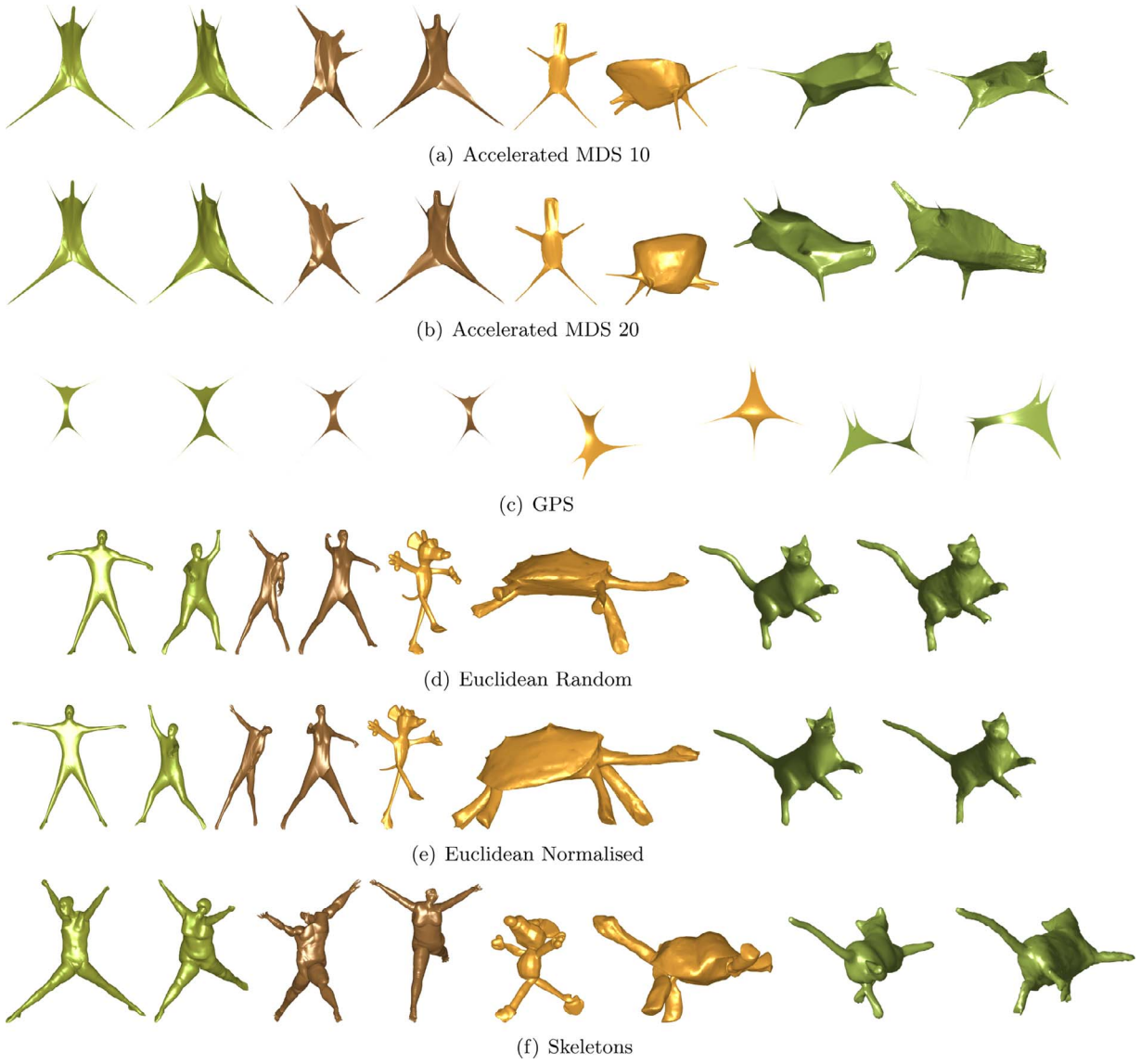


Fig. 7. Two example canonical forms for several methods from Real, Synthetic, Non-rigid along with two examples from TOSCA and TOSCA with noise respectively.

In general all methods achieve a very similar result to one-another on all the five datasets, with the exception of the GPS method, which performs significantly better on both human datasets and significantly worse on the *Non-Rigid* dataset.

To explore in detail why the methods perform poorly on the *Real* and *Synthetic* datasets, Table 6 shows the fraction of nearest-neighbour errors which exhibited the same pose as the query model. This table helps to indicate how much the low retrieval performance was due to a lack of pose-invariance. We cannot produce the same table for the *Non-Rigid* dataset, as the same poses are not used across the different shape categories. These results show that for all methods, apart from GPS, at least 97% of the errors on the *Real* dataset, and 92% of the errors on the *Synthetic* dataset were caused by retrieving a model of the same pose, but incorrect shape, for the CM-BOF retrieval framework. Slightly lower errors are caused by pose for the 3DSP retrieval framework. This is evidence that the canonical forms are not completely pose invariant. The low retrieval results on both human datasets are therefore likely due to the variation in pose within each class, even after computing canonical forms, being greater than the variation in shape between classes. The GPS method has a significantly lower error caused by lack of pose-invariance, especially for the *Synthetic* dataset. This indicates that the GPS method may be significantly more pose invariant, which probably accounts for the higher retrieval score on these two datasets. On the *Non-Rigid* dataset, where pose-invariance appears less important, it may be the high level of distortion which causes the lower retrieval performance.

5.4. Timings

The run-time of each canonical form method is shown in Table 7. The MDS methods which include the computation of a full geodesic distance matrix were run on meshes simplified to approximately 2000 vertices using MeshLab [28]. This is because the run-time of these methods on the full-resolution meshes is impractical (up to several hours per mesh). Caution is therefore needed when comparing the timings between these methods and the others.

The GPS method achieved the fastest run-time, but achieved mixed retrieval results.

The next fastest method for the full-resolution meshes is Accelerated MDS. Both values for the number of distance matrix columns used

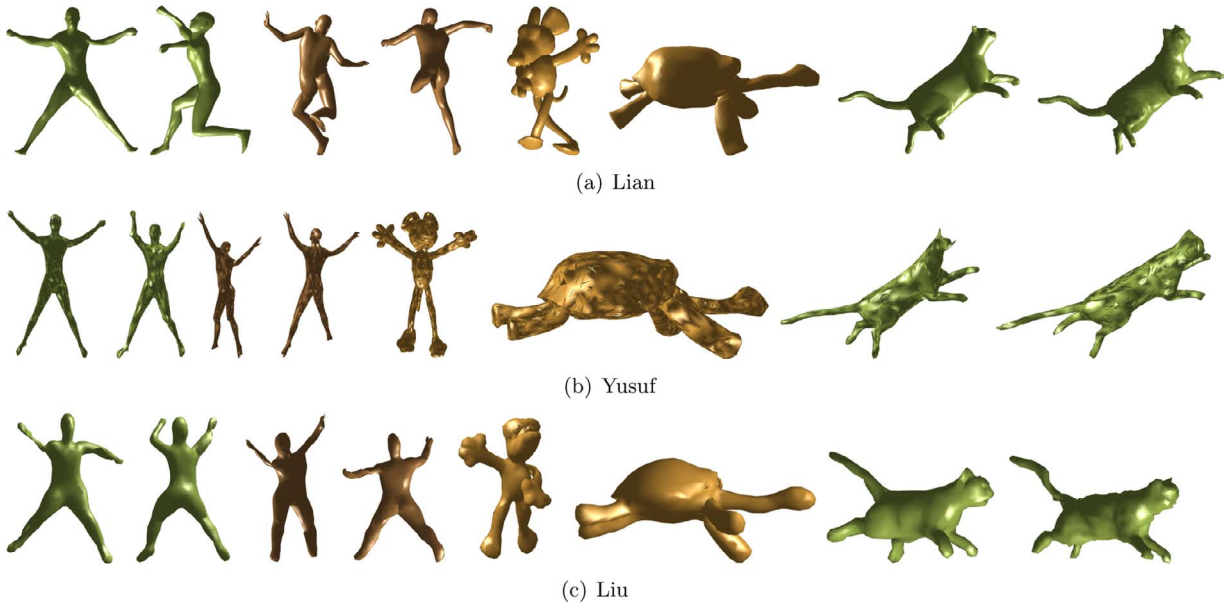


Fig. 8. Two example canonical forms for several methods from Real, Synthetic, Non-rigid along with two examples from TOSCA and TOSCA with noise respectively.

produce faster timings than all the other methods, except GPS. When compared to the standard Classic MDS method, Accelerated MDS exhibits a small drop in performance, but still achieves decent results.

The next best timings are achieved by the Constrained MDS and Skeletons methods. The Skeletons method achieved better retrieval results for the *Non-Rigid* dataset, when using the better performing CM-BOF retrieval framework.

The slowest methods run on the full-resolution meshes were the Euclidean-based methods, with the Euclidean Normalised method achieving a faster run-time. This method takes over 10 min per mesh for the *Synthetic* dataset, but may still be practical for many application where the descriptor can be computed during offline preprocessing.

6. Discussion and conclusion

Canonical forms can be useful for non-rigid retrieval, adding some pose-invariance to a rigid retrieval framework. Our experiments have shown however, that the pose-invariance is not perfect and this can significantly reduce retrieval performance on datasets where small differences in shape are important.

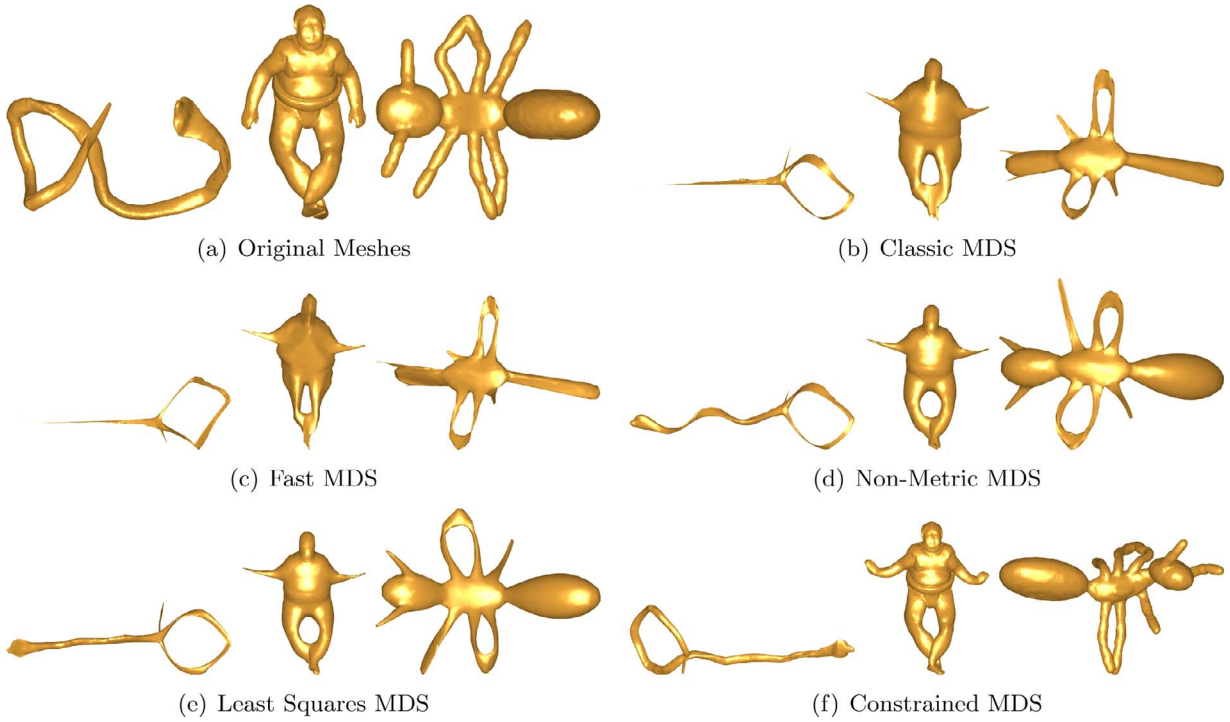


Fig. 9. Three example canonical forms produced by several methods, for meshes with topological errors from the *Non-Rigid* dataset.

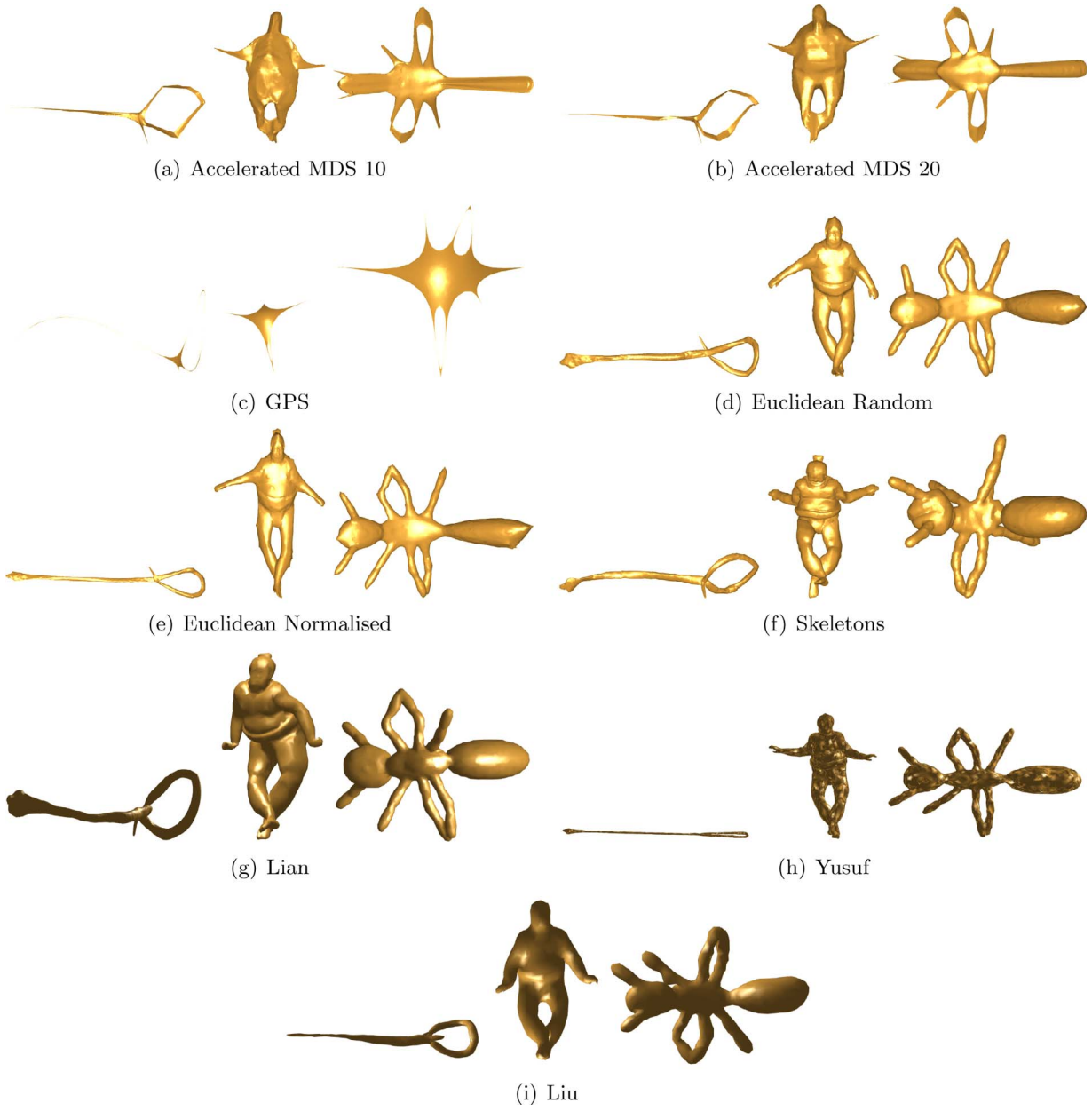


Fig. 10. Three example canonical forms produced by several methods, for meshes with topological errors from the *Non-Rigid* dataset.

Table 1
Retrieval results for the *Real* human dataset.

	CM-BOF				3DSP			
Method	NN	FT	ST	DCG	NN	FT	ST	DCG
Original Meshes	0.00	0.00	0.00	0.26	0.00	0.00	0.00	0.25
Classic MDS	0.01	0.03	0.07	0.28	0.01	0.02	0.04	0.27
Fast MDS	0.00	0.02	0.04	0.27	0.00	0.02	0.04	0.27
Non-metric MDS	0.02	0.04	0.08	0.30	0.01	**0.04**	**0.07**	**0.29**
Least Squares MDS	0.00	0.00	0.01	0.26	0.00	0.01	0.02	0.26
Constrained MDS	0.00	0.01	0.03	0.27	**0.04**	0.03	0.06	0.28
Accelerated MDS 10	0.01	0.03	0.07	0.29	0.03	0.03	0.05	0.28
Accelerated MDS 20	0.00	0.02	0.06	0.28	0.02	0.03	0.06	0.28
GPS	**0.07**	**0.06**	**0.12**	**0.33**	**0.04**	0.03	0.05	0.28
Euclidean Random	0.00	0.00	0.01	0.26	0.00	0.01	0.03	0.27
Euclidean Normalised	0.00	0.00	0.01	0.26	0.01	0.01	0.03	0.27
Skeletons	0.01	0.01	0.02	0.27	0.02	0.01	0.03	0.27
Lian	0.00	0.00	0.01	0.26	0.01	0.01	0.02	0.26
Yusuf	0.00	0.01	0.03	0.28	0.01	0.02	0.04	0.28
Liu	0.01	0.02	0.04	0.28	0.02	0.03	0.05	0.28

Table 2
Retrieval results for the *Synthetic* human dataset.

	CM-BOF				3DSP			
Method	NN	FT	ST	DCG	NN	FT	ST	DCG
Original Meshes	0.01	0.07	0.18	0.40	0.01	0.06	0.15	0.40
Classic MDS	0.10	0.22	0.39	0.54	0.16	0.15	0.25	0.48
Fast MDS	0.14	0.20	0.35	0.53	0.18	0.13	0.23	0.47
Non-metric MDS	0.09	**0.24**	**0.41**	0.55	**0.22**	**0.18**	**0.29**	**0.52**
Least Squares MDS	0.01	0.13	0.31	0.45	0.02	0.09	0.21	0.43
Constrained MDS	0.04	0.14	0.25	0.46	0.10	0.11	0.21	0.45
Accelerated MDS 10	0.10	0.20	0.36	0.53	0.17	0.13	0.23	0.47
Accelerated MDS 20	0.08	0.19	0.35	0.52	0.15	0.13	0.23	0.47
GPS	**0.40**	0.20	0.32	**0.56**	0.10	0.10	0.19	0.44
Euclidean Random	0.00	0.08	0.23	0.42	0.03	0.10	0.20	0.43
Euclidean Normalised	0.01	0.09	0.21	0.42	0.03	0.09	0.19	0.42
Skeletons	0.01	0.14	0.32	0.46	0.04	0.10	0.20	0.43
Lian	0.01	0.11	0.24	0.44	0.02	0.08	0.16	0.41
Yusuf	0.04	0.18	0.34	0.49	0.07	0.13	0.26	0.46
Liu	0.16	0.18	0.31	0.51	0.14	0.12	0.21	0.46

Table 3
Retrieval results for the *Non-Rigid* dataset.

	CM-BOF				3DSP			
Method	NN	FT	ST	DCG	NN	FT	ST	DCG
Original Meshes	0.98	0.73	0.84	0.93	0.85	0.44	0.56	0.76
Classic MDS	0.97	0.73	0.83	0.92	0.83	0.45	0.57	0.77
Fast MDS	0.94	0.65	0.77	0.88	0.71	0.35	0.47	0.69
Non-metric MDS	**0.99**	0.85	0.93	0.96	0.91	**0.57**	**0.70**	**0.84**
Least Squares MDS	**0.99**	0.86	**0.94**	**0.97**	0.84	0.47	0.61	0.78
Constrained MDS	**0.99**	0.82	0.90	0.96	**0.92**	0.56	0.68	**0.84**
Accelerated MDS 10	0.94	0.69	0.78	0.89	0.67	0.33	0.45	0.67
Accelerated MDS 20	0.95	0.71	0.81	0.91	0.74	0.37	0.49	0.70
GPS	0.75	0.45	0.58	0.75	0.54	0.22	0.33	0.58
Euclidean Random	0.98	0.77	0.87	0.94	0.80	0.43	0.56	0.75
Euclidean Normalised	0.98	0.79	0.88	0.94	0.78	0.43	0.56	0.74
Skeletons	**0.99**	0.84	0.93	0.96	0.87	0.50	0.65	0.81
Lian	**0.99**	0.84	0.93	0.96	0.85	0.48	0.61	0.78
Yusuf	0.98	**0.87**	0.93	0.96	0.88	0.51	0.65	0.81
Liu	0.96	0.75	0.84	0.92	0.76	0.38	0.50	0.71

Table 4
Retrieval results for the *TOSCA* dataset.

	CM-BOF				3DSP			
Method	NN	FT	ST	DCG	NN	FT	ST	DCG
Original Meshes	0.74	0.50	0.71	0.76	0.54	0.37	0.55	0.66
Classic MDS	0.74	0.54	0.80	0.80	0.45	0.36	0.65	0.65
Fast MDS	0.73	0.52	0.77	0.77	0.35	0.31	0.52	0.60
Non-metric MDS	0.76	0.67	0.87	0.85	0.55	0.40	0.66	0.69
Least Squares MDS	0.79	0.63	0.86	0.84	0.53	0.38	0.61	0.67
Constrained MDS	**0.88**	**0.71**	**0.89**	**0.89**	**0.68**	**0.53**	**0.78**	**0.77**
Accelerated MDS 10	0.80	0.55	0.80	0.81	0.59	0.38	0.63	0.68
Accelerated MDS 20	0.69	0.55	0.79	0.80	0.45	0.36	0.64	0.66
GPS	0.71	0.52	0.72	0.76	0.48	0.23	0.46	0.59
Euclidean Random	0.63	0.48	0.63	0.73	0.45	0.32	0.51	0.60
Euclidean Normalised	0.64	0.50	0.63	0.74	0.39	0.31	0.47	0.61
Skeletons	0.78	0.62	0.85	0.84	0.65	0.40	0.62	0.69
Lian	0.76	0.59	0.84	0.81	0.51	0.37	0.59	0.64
Yusuf	**0.88**	0.65	0.86	0.85	0.58	0.41	0.64	0.69
Liu	0.78	0.57	0.78	0.80	0.59	0.44	0.65	0.71

Table 5
Retrieval results for the *TOSCA with noise* dataset.

	CM-BOF				3DSP			
Method	NN	FT	ST	DCG	NN	FT	ST	DCG
Original Meshes	0.70	0.48	0.68	0.74	0.61	0.37	0.51	0.65
Classic MDS	0.73	0.58	0.81	0.80	0.40	0.32	0.59	0.60
Fast MDS	0.70	0.50	0.75	0.77	0.28	0.28	0.51	0.57
Non-metric MDS	0.70	0.61	**0.87**	0.83	0.50	**0.44**	**0.70**	**0.70**
Least Squares MDS	0.85	0.64	0.84	0.84	0.49	0.37	0.59	0.66
Constrained MDS	0.83	0.54	0.75	0.80	0.46	0.36	0.58	0.66
Accelerated MDS 10	0.78	0.54	0.79	0.80	0.46	0.32	0.57	0.63
Accelerated MDS 20	0.78	**0.67**	0.86	**0.85**	0.56	0.39	0.68	0.68
GPS	0.75	0.54	0.74	0.79	0.50	0.27	0.47	0.59
Euclidean Random	0.56	0.43	0.58	0.69	0.40	0.28	0.43	0.58
Euclidean Normalised	0.69	0.50	0.66	0.75	**0.64**	0.38	0.56	0.67
Skeletons	0.70	0.59	0.84	0.81	0.50	0.38	0.58	0.66
Lian	0.79	0.60	0.83	0.82	0.51	0.35	0.56	0.65
Yusuf	**0.88**	0.65	0.86	**0.85**	0.59	0.38	0.63	0.67
Liu	0.70	0.51	0.67	0.75	0.56	0.40	0.58	0.67

The majority of canonical form methods produced very similar retrieval results, but with significant variation in computational efficiency. On the *Non-Rigid* dataset, the more efficient methods produced a slightly lower retrieval result than the very expensive Least Squares MDS method. We believe however, that the large difference in efficiency may outweigh the small difference in retrieval performance for many applications.

Table 6
Fraction of incorrect nearest-neighbour retrieval results which are models with the same pose as the query.

	CM-BOF		3DSP	
Method	*Real*	*Synthetic*	*Real*	*Synthetic*
Original Meshes	1.00	0.99	1.00	1.00
Classic MDS	0.98	0.98	0.92	0.75
Fast MDS	1.00	0.92	0.84	0.41
Non-metric MDS	0.99	0.97	0.91	0.78
Least Squares MDS	1.00	1.00	1.00	0.97
Constrained MDS	0.98	0.96	0.78	0.71
Accelerated MDS 10	0.97	0.92	0.73	0.38
Accelerated MDS 20	1.00	0.97	0.80	0.53
GPS	0.83	0.25	0.52	0.19
Euclidean Random	1.00	0.97	0.95	0.79
Euclidean Normalised	0.99	0.98	0.95	0.86
Skeletons	1.00	0.99	0.91	0.91

Table 7
Average run-time per model of each canonical form method. Please note that the different methods may have been implemented in different languages and were tested on different hardware, therefore any small differences in timings are not directly comparable. The timings have been rounded to the nearest second, unless the runtime is under one second. Key: s = seconds, m = minutes.

Method	Simplification (vertices)	*Real*	*Synthetic*	*Non-Rigid*
Classic MDS	2000	45 s	47 s	45 s
Fast MDS	2000	44 s	46 s	44 s
Non-metric	2000	92 s	91 s	104 s
Least Squares MDS	2000	65 s	66 s	66 s
Constrained MDS	—	19 s	99 s	10 s
Accelerated MDS 10	—	0.6 s	4 s	0.6 s
Accelerated MDS 20	—	2 s	7 s	1 s
GPS	—	0.2 s	1 s	0.2 s
Euclidean Random	—	49 s	10 m, 49 s	23 s
Euclidean Normalised	—	41 s	7 m, 57 s	23 s
Skeletons	—	13 s	94 s	11 s
Lian	5000	19 m, 59 s	18 m, 43 s	20m 2s
Yusuf	1500	23 s	31 s	32 s
Liu	1500	8 m, 23 s	8 m, 12 s	8m 34

Our experiments have shown that different methods work better on different data. Some datasets require a greater emphasis on the pose-invariance of the canonical form, whereas other datasets require that the canonical forms cause less distortion to the original shape.

Acknowledgements

This work was supported by EPSRC Research Grant EP/J02211X/1, National Natural Science Foundation of China Grant 61300135, Hong Kong Scholars Program Grant XJ2014058, Doctoral Fund of Ministry of Education of China Grant 20130172120001, Natural Science Foundation of Guangdong Province Grant S2013040016930, Open Research Fund of State Key Laboratory Grant I3I03, and TUBITAK under the project EEEAG-115E471.

References

[1] O.K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, Skeleton extraction by mesh contraction, SIGGRAPH '08, ACM, 2008, pp. 44:1–44:10.
[2] R.A. Baeza-Yates, B.A. Ribeiro-Neto, Modern Information Retrieval - The Concepts and Technology Behind Search, Second edition, Pearson Education Ltd., 2011.
[3] M. Ben-Chen, C. Gotsman, Characterizing shape using conformal factors, Proceedings of the 1st Eurographics conference on 3D Object Retrieval, EG 3DOR'08, Eurographics Association, 2008, pp. 1–8.
[4] A.M. Bronstein, M.M. Bronstein, R. Kimmel, Numerical Geometry of Non-Rigid Shapes, Monographs in Computer Science, Springer, 2008.
[5] A. Bronstein, M. Bronstein, R. Kimmel, Numerical Geometry of Non-Rigid Shapes,

Springer New York, 2009.
[6] B. Bustos, H. Tabia, J. Vandeborre, R. Veltkamp, Coulomb shapes: using electrostatic forces for deformation-invariant shape representation (2014).
[7] CAESAR, 2013, (http://store.sae.org/caesar/).
[8] DAZ Studio, 2013, (http://www.daz3d.com/).
[9] V. de Silva, J. Tenenbaum, Global versus local methods for nonlinear dimensionality reduction, Proc. NIPS (2003) 721–728.
[10] V. de Silva, J. Tenenbaum, Global versus local methods for nonlinear dimensionality reduction, Proc. NIPS (2003) 721–728.
[11] A. Elad, R. Kimmel, On bending invariant signatures for surfaces, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1285–1295.
[12] A. Elad, R. Kimmel, On bending invariant signatures for surfaces, IEEE Trans. PAMI 25 (2003) 1285–1295.
[13] C. Faloutsos, K.-I. Lin, Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets, Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, SIGMOD '95, ACM, New York, NY, USA, 1995, pp. 163–174, http://dx.doi.org/10.1145/223784.223812.
[14] K. Hildebrandt, K. Polthier, M. Wardetzky, On the convergence of metric and geometric properties of polyhedral surfaces, Geometriae Dedicata 123 (1) (2006) 89–112.
[15] D.S. Hochbaum, D.B. Shmoys, A best possible heuristic for the k-center problem, Math. Oper. Res. 10 (2) (1985) 180–184.
[16] A. Jacobson, L. Kavan, O. Sorkine, Robust inside-outside segmentation using generalized winding numbers, Proc. SIGGRAPH 32 (4) (2013) 33:1–33:12.
[17] S. Katz, G. Leifman, A. Tal, Mesh segmentation using feature point and core extraction, Vis. Comput. 21 (8–10) (2005) 649–658.
[18] R. Kimmel, J.A. Sethian, Computing geodesic paths on manifolds, Proc. Natl. Acad. Sci. 95 (15) (1998) 8431–8435.
[19] J. Knopp, M. Prasad, G. Willems, R. Timofte, L. Van Gool, Hough transform and 3d surf for robust three dimensional classification, Proceedings of the 11th European Conference on Computer Vision: Part VI, ECCV'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 589–602.
[20] Y.-K. Lai, S.-M. Hu, R.R. Martin, P.L. Rosin, Fast mesh segmentation using random walks, Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, SPM '08, ACM, New York, NY, USA, 2008, pp. 183–191, http://dx.doi.org/10.1145/1364901.1364927.
[21] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H.V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, D. Vandermeulen, SHREC'11 track: shape retrieval on non-rigid 3d watertight meshes, Proceedings of the 4th Eurographics conference on 3D Object Retrieval, EG 3DOR'11, Eurographics Association, 2011, pp. 79–88.
[22] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R.A. Guler, L. Lai, C. Li, H. Li, F.A. Limberger, R. Martin, R.U. Nakanishi, A.P. Neto, L.G. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, R.C. Wilson, Non-rigid 3D Shape Retrieval, in: I. Pratikakis, M. Spagnuolo, T. Theoharis, L.V. Gool, R. Veltkamp (Eds.), Eurographics Workshop on 3D Object Retrieval, The Eurographics Association, 2015.
[23] Z. Lian, A. Godil, X. Sun, J. Xiao, CM-BOF: visual similarity-based 3d shape retrieval using clock matching and bag-of-features, Mach. Vis. Appl. (2013) 1–20.
[24] Z. Lian, A. Godil, J. Xiao, Feature-Preserved 3D Canonical Form, Kluwer Academic Publishers, 2013.
[25] Z. Lian, P. Rosin, X. Sun, Rectilinearity of 3d meshes, Int. J. Comput. Vis. 89 (2–3) (2010) 130–151.
[26] J. Liu, Z. Lian, J. Xiao, 3D mesh unfolding via semidefinite programming, in: I. Pratikakis, F. Dupont, M. Ovsjanikov (Eds.), Eurographics Workshop on 3D Object Retrieval, The Eurographics Association, 2017, , http://dx.doi.org/10.2312/3dor.20171059.
[27] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[28] MeshLab, 2014, (http://meshlab.sourceforge.net/).
[29] D. Pickup, X. Sun, P.L. Rosin, R.R. Martin, Non-rigid 3d human models, (https://doi.org/10.17035/d.2015.100097).
[30] D. Pickup, X. Sun, P.L. Rosin, R.R. Martin, Z. Cheng, Z. Lian, M. Aono, A. Ben Hamza, A. Bronstein, M. Bronstein, S. Bu, U. Castellani, S. Cheng, V. Garro, A. Giachetti, A. Godil, J. Han, H. Johan, L. Lai, B. Li, C. Li, H. Li, R. Litman, X. Liu, Z. Liu, Y. Lu, A. Tatsuma, J. Ye, SHREC'14 track: shape retrieval of non-rigid 3d human models, Proceedings of the 7th Eurographics workshop on 3D Object Retrieval, EG 3DOR'14, Eurographics Association, 2014.
[31] D. Pickup, X. Sun, P.L. Rosin, R.R. Martin, Z. Cheng, Z. Lian, M. Aono, A.B. Hamza, A. Bronstein, M. Bronstein, S. Bu, U. Castellani, S. Cheng, V. Garro, A. Giachetti, A. Godil, L. Isaia, J. Han, H. Johan, L. Lai, B. Li, C. Li, H. Li, R. Litman, X. Liu, Z. Liu, Y. Lu, L. Sun, G. Tam, A. Tatsuma, J. Ye, Shape retrieval of non-rigid 3d human models, Int. J. Comput. Vis. (2016) 1–25, http://dx.doi.org/10.1007/s11263-016-0903-8.
[32] D. Pickup, X. Sun, P.L. Rosin, R.R. Martin, Euclidean-distance-based canonical forms for non-rigid 3d shape retrieval, Pattern Recognit. 48 (8) (2015) 2500–2512, http://dx.doi.org/10.1016/j.patcog.2015.02.021.
[33] D. Pickup, X. Sun, P.L. Rosin, R.R. Martin, Skeleton-based canonical forms for non-rigid 3d shape retrieval, Comput. Vis. Media (2016) 1–13, http://dx.doi.org/10.1007/s41095-016-0045-5.
[34] D. Pickup, X. Sun, P.L. Rosin, R.R. Martin, Z. Cheng, S. Nie, L. Jin, Canonical forms for non-rigid 3D shape retrieval, in: I. Pratikakis, M. Spagnuolo, T. Theoharis, L.V. Gool, R. Veltkamp (Eds.), Eurographics Workshop on 3D Object Retrieval, The Eurographics Association, 2015, , http://dx.doi.org/10.2312/3dor.20151063.
[35] C. Redondo-Cabrera, R.J. López-Sastre, J. Acevedo-Rodríguez, S. Maldonado-Bascón, Recognizing in the depth: selective 3d spatial pyramid matching kernel for object and scene categorization, Image Vis. Comput. (32) (2014) 965–978.
[36] R.M. Rustamov, Laplace-Beltrami eigenfunctions for deformation invariant shape representation, Proceedings of the 5th Eurographics symposium on Geometry processing, Eurographics Association, 2007, pp. 225–233.
[37] Y. Sahillioğlu, A shape deformation algorithm for constrained multidimensional scaling, Comput. Graphics 53, Part B (2015) 156–165, http://dx.doi.org/10.1016/j.cag.2015.10.003.
[38] Y. Sahillioğlu, A shape deformation algorithm for constrained multidimensional scaling, Comput. Graphics 53 (2015) 156–165.
[39] Y. Sahillioğlu, L. Kavan, Detail-preserving mesh unfolding for nonrigid shape retrieval, ACM Trans. Graphics (TOG) 35 (3) (2016) 27.
[40] G. Shamai, M. Zibulevsky, R. Kimmel, Accelerating the computation of canonical forms for 3D nonrigid objects using multidimensional scaling, Eurographics Workshop on 3D Object Retrieval, The Eurographics Association, 2015, http://dx.doi.org/10.2312/3dor.20151057.
[41] P. Shilane, P. Min, M. Kazhdan, T. Funkhouser, The princeton shape benchmark, Proceedings of Shape Modeling Applications. (2004), pp. 167–178.
[42] K.Q. Weinberger, L.K. Saul, Unsupervised learning of image manifolds by semidefinite programming, Int. J. Comput. Vis. 70 (1) (2006) 77–90.
[43] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, Proceedings of the 14th Annual Conference on Neural Information Processing Systems, (2001), pp. 682–688.
[44] H.-B. Yan, S.-M. Hu, R. Martin, Y.-L. Yang, Shape deformation using a skeleton to drive simplex transformations, Vis. Comput. Graphics, IEEE Trans. 14 (3) (2008) 693–706.