EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Shape Decomposition using Modal Analysis

Qi-Xing Huang[1,2] Martin Wicke[1,2] Bart Adams[1,3] Leonidas Guibas[1]

[1] Stanford University [2] Max Planck Center for Visual Computing and Communication [3] Katholieke Universiteit Leuven

Abstract

We introduce a novel algorithm that decomposes a deformable shape into meaningful parts requiring only a single input pose. Using modal analysis, we are able to identify parts of the shape that tend to move rigidly. We define a deformation energy on the shape, enabling modal analysis to find the typical deformations of the shape. We then find a decomposition of the shape such that the typical deformations can be well approximated with deformation fields that are rigid in each part of the decomposition. We optimize for the best decomposition, which captures how the shape deforms. A hierarchical refinement scheme makes it possible to compute more detailed decompositions for some parts of the shape.

Although our algorithm does not require user intervention, it is possible to control the process by directly changing the deformation energy, or interactively refining the decomposition as necessary. Due to the construction of the energy function and the properties of modal analysis, the computed decompositions are robust to changes in pose as well as meshing, noise, and even imperfections such as small holes in the surface.

Categories and Subject Descriptors (according to ACM CCS): Computational Geometry and Object Modeling [I.3.5]: Geometric Algorithms, Languages, and Systems—Computational Geometry and Object Modeling [I.3.5]: Physically-Based Modeling—Simulation and Modeling [I.6.5]: Model Development—

1. Introduction

Decomposing 3D shapes into meaningful parts is a challenging problem. Segmentation algorithms for 3D geometry have wide ranging applications in various branches of computer graphics. In modeling, meaningful shape decompositions allow us to composite new shapes in an intuitive way [FKS*04]. In computer animation, new poses can be created by compositing transformed parts of a given shape [SZT*07, BP07]. Moreover, shape segmentations give us a way to understand [HOP*05], or compare and match shapes [GCO06]. In these algorithms, the segmentation provides knowledge about the semantics of the shape: Good shape decompositions should partition the shape reflecting functional or logical units. This semantic component makes the problem a hard one, and in many cases, an ill-posed one.

We are particularly interested in decomposing an object into (almost) rigid components. Previous approaches [JT05, SY07] rely on the input of multiple example poses and accurate correspondences across these example poses. However, such data sets are hard to acquire in practice, in particular if the shapes are scanned from real-world objects.

In this paper, we introduce a novel approach to shape decomposition that is based on analysis of the typical deformations of the shape $\mathcal{S}$. We aim to find a partition of the shape into parts $\mathcal{P}_i$, such that if the parts were rigid components of an articulated shape $\mathcal{A}$, the typical deformations of $\mathcal{S}$ can be approximated by poses of the articulated shape $\mathcal{A}$.

Our framework for shape decomposition and skeleton extraction is based on the observation that we can extract information about the typical deformations of a shape from the shape alone. We therefore only require a single pose to compute such a partition. Using *modal analysis*, we compute the typical deformation modes of the shape. Modal analysis uses spectral analysis of the Hessian of a deformation energy to find the shape's *eigenmodes*, which form a basis for the space of possible deformations. By using only the eigenmodes corresponding to the lowest eigenvalues, we can restrict our analysis to the low-energy, low-frequency deformations that are particularly interesting for decomposition purposes.

We then find a decomposition of the shape by minimizing the difference between an optimal articulated (piecewise

rigid) deformation defined on the parts of the decomposition and each basis vector of the space of typical deformations. The decomposition that minimizes this approximation error is our desired result.

We also introduce a method to compute subspaces of deformations that locally optimize quality measures defined on the shape. Using this method, we can compute optimized part boundaries by considering a subspace of deformations that optimizes rigidity within the adjacent parts. The same technique is used to refine the decomposition in a hierarchical fashion.

Given the output of our decomposition algorithm, we compute an articulated skeleton of the shape as a step towards an animated articulated model.

Our algorithm computes good decompositions fully automatically, and in fact the examples in this paper were computed with the same parameters, unless otherwise indicated. Nevertheless, our algorithm is highly controllable, either by modifying the energy function (e. g. by varying stiffness), or by manually controlling where and how far to refine.

Our results indicate that the decompositions computed by our algorithm nicely capture the deformations of the input shapes. Our method is very robust to different poses, surface noise, and even handles small imperfections such as holes in the surface.

The remainder of this paper is structured as follows: After discussing related work in Section 2, Section 3 will give an overview of our pipeline. In Section 4, we describe how the shape is analyzed using modal analysis. Section 5 describes how to compute a decomposition from the eigenmodes of the shape. Methods to improve the initial decomposition by boundary optimization and hierarchical refinement are presented in Section 6. Section 7 treats skeleton extraction. Our results are discussed in Section 8.

2. Related Work

Shape segmentation has been treated in scores of publications in the past. For a good overview of the literature, we refer the reader to [Sha06]. In this paper, we will restrict the discussion to papers on logical segmentations or shape decompositions, excluding work on patch segmentations used for parameterization, rendering, or mesh processing.

Probably the most robust way of performing articulated shape decomposition is to start from multiple registered example poses of a given object [JT05, SY07]. Example poses provide rich information about which points move rigidly together. However, obtaining complete, registered poses is hard in some applications, such as the segmentation of a scanned real-world model.

Given an articulated object, some of the general shape segmentations [KT03, YLL*05, LZHM06, LLS*05, KJS07, dGGV08] that use local feature descriptors are able to segment some of the rigid parts. However, these methods compute segmentations based on geometric features on the surface. These features often lie on the boundaries of logical parts, and thus such algorithms can find functional units in a shape. Noise, as well as differences in pose are however likely to distort the resulting segmentations, as purely geometric algorithms have no notion of the actual deformations a shape undergoes.

Katz et al. [KLT05] compute segmentations based on a space embedding of the shape using multi-dimensional scaling. The embedding depends on the geodesic distances between surface points and is therefore independent of pose. The method relies on finding the core component of the shape, which works well for bodies of animals and humans, but is harder to define for more general objects. Decomposition based on the shape diameter function [SSCO08] also yields segmentations that are mostly independent of pose.

Mortara et al. [MPS*04] provide an interesting approach to identifying tubular parts of a shape. Their approach works even for complicated topologies, however, it is limited to tubular segments. [LKA06] propose a method for segmentation based on the quality of the skeleton induced by the decomposition. This yields particularly nice skeletons, but objects with complex topology can cause trouble. Another interesting way of computing a skeleton is by topology-preserving mesh contraction [ATC*08].

Variational shape segmentation techniques aim to segment a shape into patches that have some common property. For example, the points within a patch might be well approximated by a parametric surface [CSAD04, WK05, YLW06] or kinematic surface [GG04, HOP*05].

Our method is most closely related to spectral techniques. Spectral techniques have been used in image segmentation and clustering (see e. g. [Wei99]). For 3D shapes, spectral analysis of the Laplacian has been used in shape segmentation, and promising results were presented [LZ07, Rus07]. Our work provides some background on why the Laplacian yields good results: The weighted Laplacian is the Hessian of a linearized thin plate energy.

Modal analysis is a technique which aims to find typical vibration modes in physical models. In the context of computer graphics, it has mainly been used to speed up computations by omitting higher-energy modes, and reducing the problem to a lower-dimensional one (see e. g. [PW89, JP02, CK05, BJ05]). Besides its obvious uses in physical modeling, modal analysis has been used in image matching [SP95], where it proves useful due to its invariance properties. We exploit these same properties to achieve robustness to noise.

3. Overview

The complete pipeline of our algorithm is illustrated in Fig. 1. We harness modal analysis, which computes the vibration modes of a deformable model, forming a basis for all possible deformations. The knowledge about how the shape

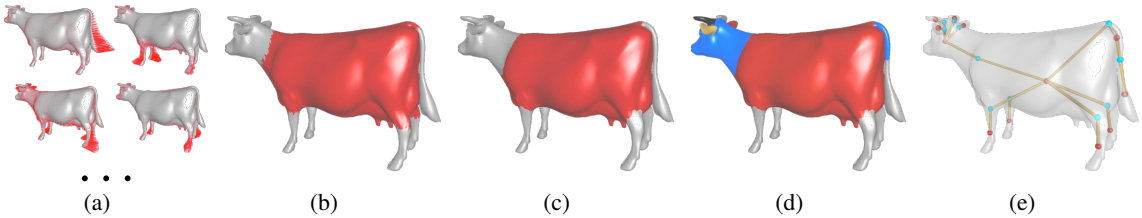


Figure 1: *Our shape decomposition pipeline. (a) Computing a space of typical deformation modes. (b) Identifying the number of segments and computing a rough initial segmentation. (c) Optimizing patch boundaries using discriminating deformations. (d) Hierarchical refinement. (e) Skeleton extraction. Note that non-adjacent patches are distinct, even if they share a color.*

deforms is encapsulated in a deformation energy. The definition of this deformation energy is entirely left to us; we use the energy function introduced in [SA07].

Modal analysis is a technique based on eigendecomposition of the Hessian of the energy function. The eigenvalues of the Hessian, the *eigenmodes* of the shape, provide a basis for the space of possible deformations. The eigenvalue associated with each eigenmode contains its respective energy content. The deformations that we are interested in lie in the space spanned by low-energy modes. These are the first deformations to appear when the shape is subjected to external forces, making them the most likely and most common deformations. When used in physics-based modeling, modal analysis used to restrict deformations to the low-energy modes since they are sufficient to represent common motions (see e. g. [PW89, BJ05]). Fig. 2 shows the first few eigenmodes of the cow model for our chosen energy. The deformation modes are represented as per-vertex displacements.

Given a basis of the space of typical deformations, we compute an *articulation score* for each of the eigenmodes. We use this score to weight the contribution of eigenmodes to the objective functions, giving more weight to more articulated eigenmodes. Since our objective functions measure how well eigenmodes can be approximated by piecewise rigid functions, this weighting makes sure that we approximate articulated deformations well, instead of diluting our results by trying to approximate deformations that cannot be well approximated by piecewise rigid functions.

Using Lloyd clustering with seed points in regions of high local rigidity, we then compute a partitioning of the shape that minimizes the weighted approximation error for all low-energy eigenmodes. In other words, the result of this optimization is a partitioning that is best suited to approximate all low-energy deformations if we consider its parts rigid.

We can improve the results of our decomposition by refining the part boundaries. For each boundary between a set of parts, we apply the same clustering technique, but we use a subspace of the typical deformations that are particularly well-suited to distinguish between a specific set of adjacent parts.

The same technique can be applied to hierarchical refinement: by computing a subspace of the low-energy deformations that is optimal for a specific part $\mathcal{P}$ of the decomposition, we can decompose $\mathcal{P}$ the same way we partitioned the original shape. The eigenvalues of the restricted eigenproblem give us a natural termination criterion.

Once the decomposition is complete, we extract a skeleton by associating each part with a rigid bone. A minimum spanning tree on the connectivity graph of the decomposition is used to connect the bones with joints. A side product of our algorithm are association weights that tether the surface to the skeleton. While these weights are no replacement for rigging weights created by a skilled animator, they are a good starting point for rigging the model.

4. Deformation Analysis

In order to analyze the deformations of a surface $\mathcal{S}$, let us consider a discretized version of $\mathcal{S}$ defined by a number of points $\mathcal{P} = \{\mathbf{p}_1 \dots \mathbf{p}_N\}$. The amount of energy necessary to deform the shape is given by an energy function $E(\mathbf{u})$, where the deformation $\mathbf{u} = [\mathbf{u}_1^T \dots \mathbf{u}_N^T]^T$ consists of displacement vectors for each of the defining points in $\mathcal{P}$.

Modal analysis [PW89] analyzes the Hessian $\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{u}^2}$ of the deformation energy to infer knowledge about the deformations of $\mathcal{S}$: If we consider the surface to be a dynamic system, the eigenspectrum of $\mathbf{H}$ gives us information about the vibration modes (in the eigenvectors $\mathbf{v}^k$) and vibration frequencies of $\mathcal{S}$ (in the eigenvalues λ_k). Fig. 2 shows an example using the energy function defined below. The eigenvalues indicate the energy content of the corresponding deformation mode. Therefore, we are mainly interested in eigenvectors to small eigenvalues. Since these modes have low energy content, they are the modes that the shape most likely undergoes: Only little energy is needed to deform the shape within the space of low-energy deformations, and such deformations occur naturally. On the other hand, high-energy deformations require strong external forces, and are therefore uncommon. Note that the eigenvectors to eigenvalue 0 span the space of motions to which the deformation energy is invariant. In almost all cases, these include rigid-body motions (a six-dimensional subspace). In the following, these *trivial* eigenvectors will be ignored.

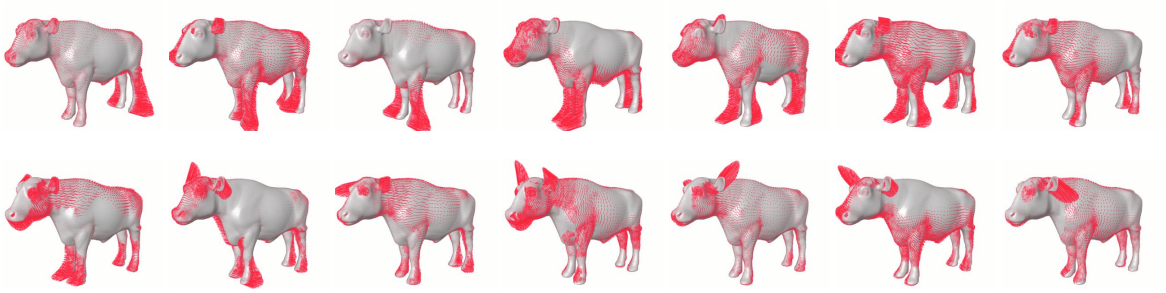


Figure 2: *The first 14 non-trivial vibration modes of the Cow2 model using the energy (1). They nicely capture the articulated structure of the shape, separating body parts such as legs, head, and ears. The deformation fields are shown as vectors $\mathbf{v}_i^k$ at each vertex $\mathbf{p}_i$.*

We will assume that the eigenvalues (and their corresponding normalized eigenvectors) are ordered in ascending order, i. e. $\forall i < j : \lambda_i \leq \lambda_j$. The space spanned by the first N_t nontrivial eigenvectors, which we will also call the space of *typical deformations*, shall be denoted as $\mathcal{U}_{N_t}$.

4.1. Deformation Energy

In order to perform modal analysis, we require an energy function. For simplicity, we use an energy defined solely by the surface, which does not require discretization of the interior of the shape. We adopt the "as rigid as possible" deformation energy proposed in [SA07], shown in (1). Other shell energies, such as a thin-shell energy [GHDS03], or an energy favoring isometric deformations [KMP07] can be used as well, with similar results. For any set of points $\mathcal{P}$, we define

$$E_{\mathcal{P}}(\mathbf{u}) = \sum_{\mathbf{p}_i \in \mathcal{P}} \min_{\mathbf{c}_i} \sum_{j \in \mathcal{N}_i} w_{ij} \|\mathbf{c}_i \times (\mathbf{p}_i - \mathbf{p}_j) - \mathbf{u}_i + \mathbf{u}_j\|^2, \quad (1)$$

Here, $\mathcal{N}_i = \mathcal{N}(\mathbf{p}_i) = \{j : \|\mathbf{p}_j - \mathbf{p}_i\| < \varepsilon\}$ is a set of indices of neighboring points for each point $\mathbf{p}_i$. The terms $\mathbf{c}_i \times (\mathbf{p}_i - \mathbf{p}_j)$ are a first-order approximation to a rotation of $\|\mathbf{c}_i\|$ rad around the axis $\mathbf{c}_i$. The best matching $\mathbf{c}_i$ are computed by shape matching to the neighborhood. The weights w_{ij} are nonzero only if $i \in N_j$ or $j \in N_i$, and should reflect sampling density and local geometry; we found that the simple symmetric $w_{ij} = (|\mathcal{N}_i| + |\mathcal{N}_j|)^{-1}$ gives good performance.

Common deformations have low energies, and are nicely represented in the space spanned by the first nontrivial eigenvectors $\mathcal{U}_{N_t} = \text{span}\{\mathbf{v}^1, \ldots, \mathbf{v}^{N_t}\}$, even for small N_t. In practice, we use $N_t \approx 100$.

4.2. Computing the Hessian

In order to perform modal analysis, we need the Hessian of the deformation energy. Since the energy function uses a shape matching step, its Hessian is hard to evaluate directly. To compute the Hessian of any energy that uses such an optimization step, we can use the following observation:

Given a function $g(\mathbf{x}) = f(\mathbf{x}, \mathbf{y}(\mathbf{x}))$, where $\mathbf{y}(\mathbf{x}) = \text{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$, the Hessian of g is given by

$$\frac{\partial^2 g}{\partial \mathbf{x}^2}(\mathbf{x}) = \frac{\partial^2 f}{\partial \mathbf{x}^2}(\mathbf{x}, \mathbf{y}(\mathbf{x})) + \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}}(\mathbf{x}, \mathbf{y}(\mathbf{x})) \cdot \left[\frac{\partial^2 f}{\partial \mathbf{y}^2}(\mathbf{x}, \mathbf{y}(\mathbf{x}))\right]^{-1} \cdot \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{x}}(\mathbf{x}, \mathbf{y}(\mathbf{x})). \quad (2)$$

Please refer to the appendix for a derivation. To apply this equality, we rewrite our energy function in matrix form:

$$E_{\mathcal{P}} = [\mathbf{u}^T \ \mathbf{c}^T] \begin{pmatrix} \mathbf{L} \otimes \mathbf{I}_3 & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{c} \end{bmatrix}, \quad (3)$$

where $\mathbf{u}$ contains displacement vectors for all points in $\mathcal{P}$, and $\mathbf{c}$ contains the optimal vectors $\mathbf{c}$ at all points in $\mathcal{P}$. Furthermore, $\mathbf{L}$ is the weighted graph Laplacian, in our case

$$\mathbf{L}_{ij} = \begin{cases} \sum_k -w_{ik} & i = j, \\ w_{ij} & \text{otherwise.} \end{cases} \quad (4)$$

$\otimes$ denotes the standard tensor product and $\mathbf{I}_3$ is the 3×3 idenitity matrix. The matrix $\mathbf{B}$ is a block matrix whose 3×3 blocks are defined as

$$\mathbf{B}_{ij} = \begin{cases} (\sum_i w_{ik}(\mathbf{p}_i - \mathbf{p}_k)) \times & j = i, \\ (-w_{ij}(\mathbf{p}_i - \mathbf{p}_j)) \times & \text{otherwise,} \end{cases} \quad (5)$$

where we use the notation $\mathbf{x}\times$ as the matrix associated with taking the cross product with $\mathbf{x}$. Finally, $\mathbf{C} = \text{diag}(\mathbf{C}_1 \ldots \mathbf{C}_{|\mathcal{P}|})$ is a block diagonal matrix that contains the covariance matrices of the point neighborhoods:

$$\mathbf{C}_i = \sum_{j \in \mathcal{N}_i} w_{ij}(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i - \mathbf{p}_j)^T. \quad (6)$$

By associating $\mathbf{u}$ with $\mathbf{x}$ and $\mathbf{c}$ with $\mathbf{y}$, we can use (2) to obtain an expression for the Hessian of $E_{\mathcal{P}}$:

$$\frac{\partial^2 E_{\mathcal{P}}}{\partial \mathbf{u}^2} = \mathbf{H}_{\mathcal{P}} = \mathbf{L} \otimes \mathbf{I}_3 - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^T, \quad (7)$$

which we in turn analyze to compute eigenmodes of the shape. Note that if we omit the shape matching part and simply compare displacements, the Hessian becomes the Laplacian. Methods based on spectral analysis of the Laplacian are therefore closely related to a special case of our algorithm for a particular choice of energy function.

5. Decomposition

Modal analysis provides us with a space of likely, or typical deformations, $\mathcal{U}_{N_t}$, as well as a basis $\mathbf{U}_{N_t} = [\mathbf{v}^1, \ldots, \mathbf{v}^{N_t}]$ for this space. We will approximate those basis vectors with piecewise rigid deformations, and use the approximation error as an objective function to compute an optimal decomposition into rigid parts. We have found that weighting the contributions of the basis vectors to the objective function greatly improves the results.

In this section, we will describe how we compute articulation scores for each basis vector, and how we cluster the vertices of our shape into parts such that the objective function is optimized. The results of this initial decomposition are further improved by boundary optimization and hierarchical refinement as described in Section 6.

5.1. Articulation Scoring

As mentioned above, we are aiming to find one decomposition that can approximate all basis vectors $\mathbf{v}^k$. However, not all basis vectors are equally important, and we have to choose weights to adjust the contribution of their respective approximation errors to the objective function.

An obvious strategy for choosing weights a_k for $\mathbf{v}^k$ is to use their corresponding eigenvalues: $a_k = 1/\sqrt{\lambda_k}$ [Rus07]. This gives higher weights to lower-energy modes.

We found that in our case, much better results are obtained using a variation: Some deformations can not be well approximated with piecewise rigid functions. In order to avoid overfitting to these non-articulated deformations during clustering, we give more weight to basis vectors that are articulated, and therefore should be easy to approximate.

To compute such weights, we first define the *local rigidity error* for a given deformation vector $\mathbf{u}$ at point i as the contribution of this point to the global energy:

$$E_i(\mathbf{u}) = \min_{\mathbf{c}_i} \sum_{j \in \mathcal{N}_i} \|\mathbf{c}_i \times (\mathbf{p}_i - \mathbf{p}_j) - \mathbf{u}_i + \mathbf{u}_j\|^2. \quad (8)$$

Assembling the square roots of the rigidity errors into a vector $\mathbf{E}(\mathbf{u}) = [\sqrt{E^1(\mathbf{u})}, \ldots, \sqrt{E^N(\mathbf{u})}]^T$, we can use the 1-norm to define a score for any deformation

$$a(\mathbf{u}) = \frac{1}{\|\mathbf{E}(\mathbf{u})\|_1} \quad (9)$$

Note that this definition is closely related to using eigenvalues, since $\sqrt{\lambda_k} = \|\mathbf{E}(\mathbf{v}^k)\|_2$. However, using the 2-norm favors deformations that are evenly distributed, while in the 1-norm, articulated deformations have lower energies.

Finally, we define the weights for the basis vectors $\mathbf{v}^k$ as

$$a_k = \frac{a(\mathbf{v}^k)}{a_{\min}}, \quad (10)$$

where $a_{\min} = \min_k a(\mathbf{v}^k)$. We use the articulation scores whenever we construct an objective function that is optimized over all basis vectors $\mathbf{U}_{N_t}$. Given an arbitrary function $f(\mathbf{u})$ defined for a deformation field, a convenient short-hand notation will be

$$f(\mathbf{U}_{N_t}) = \sum_{k=1}^{N_t} a_k f(\mathbf{v}^k) \quad (11)$$

to denote the weighted objective function considering the complete subspace $\mathcal{U}_{N_t}$.

5.2. Clustering

The decomposition partitions the shape into disjoint patches $\mathcal{P}_j$. We use a variant of k-means clustering to find an optimal decomposition. For this process, we need strategies to find a good number of clusters, locations for cluster seeds, and a local error measure to guide the clustering. We will describe these components in the following paragraphs.

Since we are aiming to approximate a deformation $\mathbf{u}$ with a piecewise rigid deformation field, we compute an optimal linearized rigid approximation $\mathbf{T}^{\mathbf{u}}_{\mathcal{P}_j}(\mathbf{p}) = \mathbf{p} + \bar{\mathbf{c}}^{\mathbf{u}}_j + \mathbf{c}^{\mathbf{u}}_j \times \mathbf{p}$ for each part $\mathcal{P}_j$ of the shape. With each part, we store such transformations for all basis vectors, $\mathbf{T}^{\mathbf{v}^k}_{\mathcal{P}_j}$, $k = 1 \ldots N_t$.

Given a deformation field $\mathbf{u}$ and the transformation $\mathbf{T}^{\mathbf{u}}_{\mathcal{P}_j}$, we can then define a *local approximation error* for any point $\mathbf{p}_i$ with respect to a part $\mathcal{P}_j$ as

$$l_{ij}(\mathbf{u}) = \|\mathbf{T}^{\mathbf{u}}_{\mathcal{P}_j}(\mathbf{p}_i) - \mathbf{p}_i - \mathbf{u}_i\|^2 = \|\bar{\mathbf{c}}^{\mathbf{u}}_j + \mathbf{c}^{\mathbf{u}}_j \times \mathbf{p}_i - \mathbf{u}_i\|^2. \quad (12)$$

We compute the optimal $\mathbf{T}^{\mathbf{u}}_{\mathcal{P}_j}$ by choosing $\bar{\mathbf{c}}^{\mathbf{u}}_j$ and $\mathbf{c}^{\mathbf{u}}_j$ to minimize $\sum_{\mathbf{p}_i \in \mathcal{P}_j} l_{ij}(\mathbf{u})$, which requires solving a linear system in a least-squares sense. In the following, we will use $l_{ij}(\mathbf{U}_{N_t})$ as defined by Eq. 11.

Since the number of parts is initially unknown, we start with only one patch, and add clusters in a greedy fashion. Whenever a patch is added, we select the new patch seed to be the surface point that has an above-average local fitting error $l_{ij}(\mathbf{U}_{N_t}) > l_{\text{avg}}$ with respect to its currently assigned part $\mathcal{P}_j$ *and* minimizes the local rigidity error $E_i(\mathbf{U}_{N_t})$. Fig. 3 shows some models colored according to local rigidity. Points with high local rigidity tend to lie near the center of rigid parts of the model, and are therefore good starting points for the clustering algorithm.

After seed points have been identified, we use a standard multi-seed breadth-first search to assign all surface points to clusters. A few iterations of Lloyd relaxation minimize the local approximation error.

To determine the optimal number of patches, we plot the logarithm of the maximum local approximation error as a function of the number of seeds n used. Fig. 4 shows examples. Note that the maximum fitting error has a spike in its (discrete) curvature whenever we find a sensible number of seeds. We therefore pick the number of seed points that maximizes the discrete curvature.

This strategy reliably finds a good number of seeds for simple models whose articulated structure is clear. For more

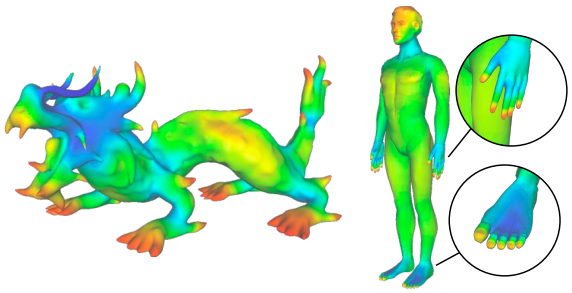
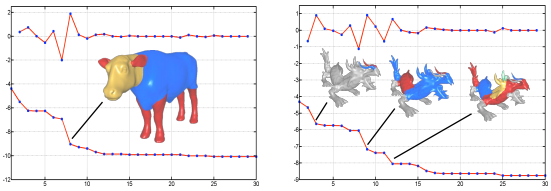


Figure 3: *The points in the models are colored according to local rigidity. The points with highest local rigidity (red) lie near the centers of rigid parts.*



(a) (b)

Figure 4: *The logarithm of the maximum local fitting error (bottom graph) plotted against the number of patches, as well as the discrete curvature of the function (top graph). Good patch counts maximize the discrete curvature.*

complicated models, the results can be ambiguous, even though all possibilities correspond to good segmentations. If no user guidance is available in an ambiguous case, we choose the number of patches that is preceded by the biggest decline in approximation error. Once hierarchical refinement is used, choosing the right number of patches become much less important. Our experiments suggest that hierarchical refinement leads to the same result independent of the number of patches chosen in the initial iteration.

6. Patch Optimization

Once we have obtained an initial decomposition, we optimize the parts further by considering deformations from $\mathcal{U}_{N_t}$ that are optimized to be particularly useful in the region of interest. We use these fields for boundary optimization as well as hierarchical refinement.

6.1. Locally Optimized Deformation Fields

Given a region of interest $\mathcal{I}$, we can compute a subspace of $\mathcal{U}_{N_t}$ that contains the deformations that have low energy content $E_{\mathcal{I}}$ within $\mathcal{I}$, while not considering their behavior in the rest of the shape. As before, we can compute the Hessian $\mathbf{H}_{\mathcal{I}}$, which is simply a restriction of $\mathbf{H}_{\mathcal{P}}$ to vertices in $\mathcal{I}$.

Analogous to computing the typical deformations of the complete shape, we now compute a subspace of the typical deformations that have small deformation energy in $\mathcal{I}$. Since we are not choosing from the complete space of possible deformations but from the subspace $\mathcal{U}_{N_t}$, this leads to a generalized eigenproblem

$$\mathbf{U}_{N_t}^T \mathbf{H}_{\mathcal{I}} \mathbf{U}_{N_t} \mathbf{x} = \kappa \mathbf{U}_{N_t}^T \mathbf{S}_{\mathcal{I}} \mathbf{U}_{N_t} \mathbf{x}, \tag{13}$$

Where the selection matrix $\mathbf{S}_{\mathcal{I}} = \mathrm{diag}(\delta_{\mathcal{I}}) \otimes \mathbf{I}_3$ is a diagonal matrix with zero entries for all points not in $\mathcal{I}$.

As before, we obtain eigenvalues κ_k and eigenvectors $\mathbf{x}^k$. The eigenvectors in turn yield deformation vectors $\mathbf{z}^k = \mathbf{U}_{N_t} \mathbf{x}^k$, which we assemble into a matrix $\mathbf{U}_{\mathcal{I}} = [\mathbf{z}^1, \ldots, \mathbf{z}^{N_t}]$.

When using the locally optimized vector fields, we also modify the articulation weights a_k to consider only rigidity within $\mathcal{I}$.

6.2. Local Boundary Optimization

In order to refine the boundaries between parts, we define a kernel region $\mathcal{K}_j$ and a support region $\mathcal{R}_j$ for each part $\mathcal{P}_j$. Suppose the maximum local approximation error of points in $\mathcal{P}_j$ is $R_{\mathcal{P}_j} = \max_{\mathbf{p}_i \in \mathcal{P}_j} l_{ij}(\mathbf{U}_{N_t})$. We define the kernel region as all points in $\mathcal{P}_j$ whose local approximation error is small,

$$\mathcal{K}_j = \{\mathbf{p}_i \in \mathcal{P}_j \mid l_{ij}(\mathbf{U}_{N_t}) < \frac{R_{\mathcal{P}_j}}{4}\}. \tag{14}$$

The support region $\mathcal{R}_j$ is the largest contiguous set of points that includes the kernel $\mathcal{K}_j$, and contains neither points that are part of another kernel, nor points with a local approximation error $l_{ij}(\mathbf{U}_{N_t}) > 2R_{\mathcal{P}_j}$. We compute $\mathcal{R}_j$ using breadth-first search.

The boundaries of patches are optimized for groups of patches whose support regions overlap. These are the smallest units that can be optimized independently. We start assembling patch groups by forming groups of two patches: Each pair of neighboring patches $\{\mathcal{P}_i, \mathcal{P}_j\}$, where $\mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset$, forms one group. If there is a three-way intersection between patches $\mathcal{P}_i$, $\mathcal{P}_j$, and $\mathcal{P}_k$, i. e. $\mathcal{R}_i \cap \mathcal{R}_j \cap \mathcal{R}_k \neq \emptyset$, we remove the two-element groups and add a group $\{\mathcal{P}_i, \mathcal{P}_j, \mathcal{P}_k\}$. This procedure is repeated for more complex intersections, although these are rare. An example of a patch group with 4 elements is shown in Fig. 5.

For each group $\mathcal{G} = \{\mathcal{P}_{j_1}, \ldots, \mathcal{P}_{j_{|\mathcal{G}|}}\}$, we compute a basis of locally optimal deformations $\mathbf{U}_{\mathcal{G}}$ as described in Section 6.1, and reapply the clustering method as described in Section 5.2 for all parts in $\mathcal{G}$. This redefines the boundaries between parts in $\mathcal{G}$, while leaving the rest of the segmentation untouched. Fig. 5 shows an example of this process.

6.3. Hierarchical Refinement

It is straightforward to extend our algorithm to support hierarchical refinement. To refine a part $\mathcal{P}_j$, we compute locally optimized typical deformations $\mathbf{U}_{\mathcal{P}_j}$ as described in Section 6.1. We then continue with the segmentation procedure as discussed in Section 5, restricting our operations to $\mathcal{P}_j$ only. Fig. 6 shows an example of hierarchical refinement on the dragon's leg. For non-interactive processing, we

Figure 5: *Optimizing the patch boundaries. (a) A group of 4 patches before the optimization. The support regions of the leg, tail, and body patches overlap in the gray area; the kernels are colored red, blue, and gold. (b) Optimized local vector fields are computed to determine the patch boundaries. (c) Result of the boundary optimization.*

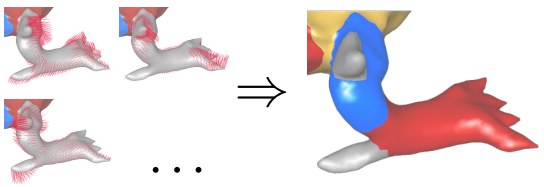


Figure 6: *A leg of the dragon is refined using deformation fields optimized for this patch.*

stop the hierarchical refinement when the smallest non-zero eigenvalue κ_1 computed for a part $\mathcal{P}_j$ is greater than the largest initial eigenvalue computed for the complete shape, λ_{N_t} (note that as we restrict $\mathcal{P}_j$, κ_1 increases). Intuitively, this criterion stops processing when we cannot find vector fields from $\mathcal{U}_{N_t}$ whose restriction to $\mathcal{P}_j$ is interesting.

Since hierarchical refinement reuses the initial eigenvectors and applies a global termination criterion, the result of hierarchical refinement is typically independent of the initial number of patches, making the "correct" number of patches a much less critical parameter.

7. Skeleton Extraction

The decomposition extracted in the previous sections can be used to define a skeleton. Each part is associated with a bone of the skeleton. We will first discuss how to compute association weights, before we extract the skeleton structure.

7.1. Association Weights

Association weights determine how the rigid transformations of the skeleton bones are interpolated onto the surface. In our case, the transformations are given for the kernel of each part. We compute weights $w(\mathcal{P}_j, \mathbf{p}_i)$ that determine the contribution of the rigid transformation given for the part $\mathcal{P}_j$ to the transformation of $\mathbf{p}_i$. We choose these weights to be local, such that only neighboring parts contribute to the deformation of a surface point.

For each point $\mathbf{p}_i$, we define a set of neighboring parts $\mathcal{G}(\mathbf{p}_i) = \{\mathcal{P}_j \mid \mathbf{p}_i \in \mathcal{R}_j\}$, and compute locally optimized typical deformation fields $\mathbf{U}_{\mathcal{G}(\mathbf{p}_i)}$.

We then find the association weights by minimizing the difference between deformations created using the skeleton and the typical vector fields. Given the (linearized) rigid transformation $\mathbf{T}^{\mathbf{u}}_{\mathcal{K}_j}$ that minimizes the maximum local approximation error $\max_{\mathbf{p}_i \in \mathcal{K}_j} l_{ij}(\mathbf{u})$ for some deformation $\mathbf{u}$, we seek weights $w(\mathcal{P}_j, \mathbf{p}_i)$ that minimize

$$\sum_k \left\| \mathbf{p}_i + \mathbf{z}_i^k - \sum_j w(\mathcal{P}_j, \mathbf{p}_i) \mathbf{T}^{\mathbf{z}_i^k}_{\mathcal{K}_j}(\mathbf{p}_i) \right\|^2, \qquad (15)$$

subject to the positivity and partition of unity. This minimization can be solved efficiently as a quadratic program.

The resulting weights faithfully reproduce the locally optimized vector fields. However, since the patch group associated with each point changes, there is no formal guarantee that the computed weights are smooth across the surface. Therefore, we apply a few steps of Laplacian smoothing, and renormalize the weights.

7.2. Skeleton Structure

While it is obvious that each part of the decomposition should be associated with one bone of the skeleton, it is unclear what is the best joint structure connecting these bones. We use the association weights to connect those bones whose associated patches overlap most. To this end, we consider a weighted neighborhood graph of the decomposition. Each edge in this graph is weighted by

$$w(\mathcal{P}_j, \mathcal{P}_k) = \left[\sum_{\mathbf{p}_i \in \mathcal{P}} w(\mathcal{P}_j, \mathbf{p}_i) w(\mathcal{P}_k, \mathbf{p}_i) \right]^{-1}. \qquad (16)$$

We then compute the joint structure of the skeleton as the minimum spanning tree (MST) of the weighted patch neighborhood graph.

We position each node of the MST with the barycenter of its associated kernel. Each edge in the MST is associated with a joint, which is positioned at the barycenter of the boundary curve between the two parts connected by this joint. Fig. 8 shows skeletons for various models.

8. Results and Discussion

We have tested our segmentation algorithm on various surface models. The results of the segmentations, as well as the extracted skeletons, are shown in Fig. 8. Table 1 summarizes timing and model statistics. All timings were measured on a 3.2 GHz PC with 2GB RAM. The total computation time is clearly dominated by the eigen-decomposition of the Hessian. This decomposition has to be performed only once per model, all other steps allow for interactive intervention.

Note that our method is applicable to noisy and even incomplete models. For demonstration purposes, we perturbed the vertex positions of the Cow2 model by adding Gaussian

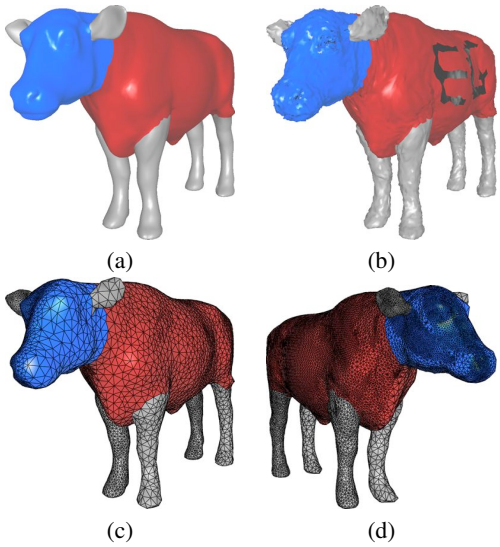


Figure 7: *Robustness to noise and sampling. (a) Decomposition of a uniformly sampled version of the Cow2 model. (b) With noise and holes added. (c),(d) Irregular sampling. The right side is sampled more densely than the left. The resulting decomposition shows no appreciable differences to the uniformly sampled version.*

noise (with a variance equal to 1% of the model diagonal) in normal direction, and carved holes in the side of its body, head, and legs. As shown in Fig. 7, the segmentation is robust against these modifications as long as the underlying articulated structure is not changed. We have also run the decomposition on a Cow2 model with varying sampling density. As long as the energy is chosen appropriately, the surface sampling has no influence on the result.

The segmentation produced by our algorithm is also robust to isometric and/or articulated deformations. The two elephant models shown in Fig. 8 (a) have significantly different poses, yet the computed segmentations are virtually identical. This is to be expected since even though the deformation energy is non-linear, the deformation modes do no change significantly after isometric deformations.

Fig. 8 (b) shows details of the decomposition of the Man model. The figure shows two possible initial decompositions (one with 5, one with 12 segments), as well as the final result of hierarchical refinement. The hierarchical refinement converges to the same solution independent of the number of initial parts. This behavior is common, although we cannot give a formal guarantee due to the properties of the seed selection and clustering algorithm.

We have computed decompositions for the Horse and Man model, which we compare to segmentations obtained using [LZ07], provided by the authors, as well as an implementation of [KT03]. These segmentations are shown in Fig. 9.

Model	N	N_c	t_g	t_c	t_b	t_h	t_w	T
Cow1	11273	11	60.2	2.1	2.4	3.1	1.1	68.9
Cow2	16914	8	96.2	1.1	3.1	—	2.1	102.5
Dragon	22502	39	170.1	4.1	4.5	6.1	5.4	190.2
Elephant1	20002	18	120.1	2.1	1.6	4.5	3.2	131.5
Elephant2	20002	18	118.7	2.1	1.7	4.3	2.8	129.8
Horse	19850	15	106.1	1.1	1.2	2.4	1.1	119.1
Man	14603	32	80.4	2.2	2.3	1.7	2.5	89.1
Raptor	22502	33	150.1	4.3	3.2	4.4	3.4	165.4

Table 1: *Model statistics and computation times. Shown are the number of surface points N, the number of clusters N_c, as well as computation times (in seconds) for the global typical vector fields t_g, clustering t_c, boundary refinement t_b, hierarchical refinement t_h, association weight computation t_w, and total time T.*

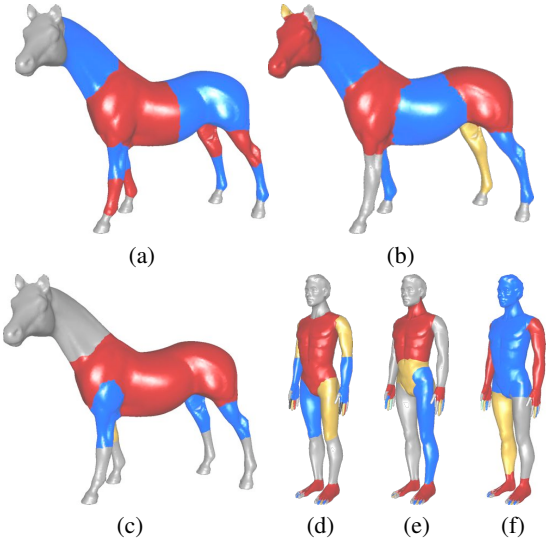
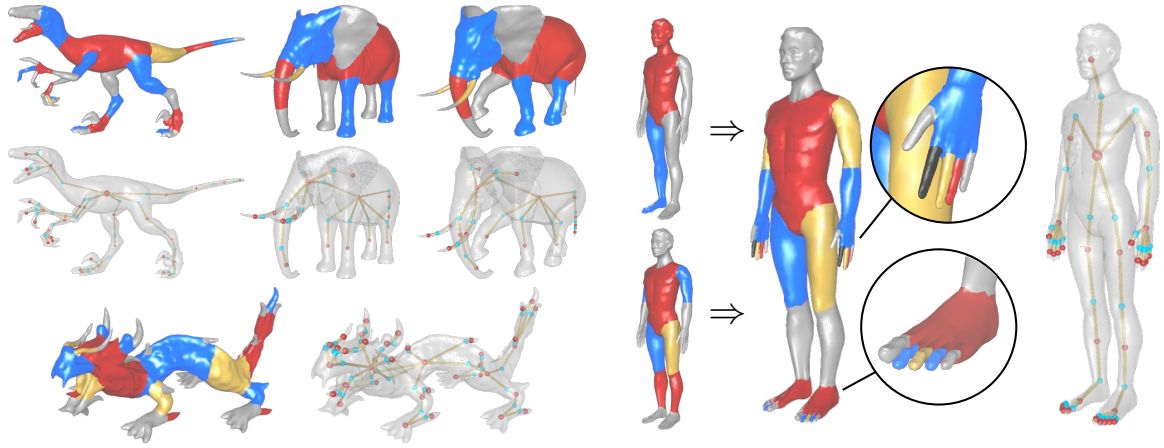


Figure 9: *(a) Segmentation of the Horse model using our method. (b) Using [LZ07]. (c) Using [KT03]. (d) Segmentation of the Man model using our method. (e) Using [LZ07]. (f) Using [KT03]*

Our method captures the articulated nature of the model. However, small features such as the horse's ears are separated late in the refinement, since their geometric structure makes them quite stable against deformations. Such features are more likely to be found by purely geometry-based algorithms, such as [LZ07].

If the desired result is not a decomposition based on the natural deformations of an object, but one based on surface texture, or small surface features, our method is of only limited use. Since our method finds a decomposition that best approximates the articulated nature of an object, the returned decomposition in such cases might not correspond to logical units.



(a) (b)

Figure 8: *(a) Decompositions and extracted skeletons for various surface models. Note that non-adjacent patches are distinct even if they share the same color. (b) The decomposition of the Man model. We can choose either 5 or 12 segments in the initial clustering step. In both cases, the hierarchical decomposition scheme terminates with the same result shown on the right.*

9. Conclusion

We have presented a shape decomposition algorithm based on optimal approximation of low-energy deformations. The method extracts the subspace of low-energy deformations for a given deformation energy. Using the information obtained by analyzing the typical deformations of the shape, we can compute meaningful shape decompositions using only a single pose, without resorting to heuristics based on geometric features.

Our method is robust to noise and even small holes in the shape. It is therefore possible to apply it directly to scanned models. We believe our method will be very useful for rapid prototyping of animated models from real-world models, for example clay models. Note also that the method is very general and can be computed to any surface, or even volumetric representation, as long as an energy function and sensible neighborhoods can be defined.

Using the skeletons extracted for each shape, it is possible to perform deformation transfer between different models. To accomplish this, it would be necessary to obtain compatible skeletons for different models, with some isomorphism describing correspondences between joints and bones in either model. In the future, we will work on forcing a specific skeleton layout in order to facilitate deformation transfer from another model.

Note that we have complete freedom in designing the deformation energy. In all examples shown in this paper we have used an energy definition that is uniform across the surface. A possible extension to our method would incorporate additional knowledge about the structure of the surface into the deformation energy, for example, by setting lower stiffness weights at joints.

Acknowledgments

This research was funded by the Max-Planck Center for Visual Computing and Communication, as well as NSF grants ITR 0205671 and FRG 0354543, NIH grant GM-072970, and DARPA grant HR0011-05-1-0007. Qi-xing Huang is supported by the Mr. and Mrs. Chin-Nan Chen Stanford Graduate Fellowship. Bart Adams is funded by the Fund for Scientific Research, Flanders (F.W.O.-Vlaanderen). We would like to thank the Stanford University and Aim@Shape for providing various models, and the anonymous reviewers for their helpful comments.

References

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton Extraction by Mesh Contraction. *ACM Transactions on Graphics 27*, 3 (2008).

[BJ05] BARBIČ J., JAMES D. L.: Real-Time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics 24*, 3 (2005), 982–990.

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics 26*, 3 (2007), 72.

[CK05] CHOI M. G., KO H.-S.: Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation. *IEEE Transactions on Visualization and Computer Graphics 11* (2005), 91–101.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. 905–914.

[dGGV08] DE GOES F., GOLDENSTEIN S., VELHO L.: A Hierarchical Segmentation of Articulated Bodies. In *Eurographics Symposium on Geometry Processing (SGP)* (2008).

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Mod-

eling by example. *ACM Transactions on Graphics 23*, 3 (2004), 652–663.

[GCO06] GAL R., COHEN-OR D.: Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics 25*, 1 (2006), 130–150.

[GG04] GELFAND N., GUIBAS L. J.: Shape Segmentation Using Local Slippage Analysis. In *Symposium on Geometry Processing* (2004), pp. 219–228.

[GHDS03] GRINSPUN E., HIRANI A. N., DESBRUN M., SCHRÖDER P.: Discrete shells. In *Symposium on Computer Animation* (2003), pp. 62–67.

[HOP*05] HOFER M., ODEHNAL B., POTTMANN H., STEINER T., WALLNER J.: 3D shape recognition and reconstruction based on line element geometry. In *IEEE International Conference on Computer Vision* (2005), vol. 2, pp. 1532–1538.

[JP02] JAMES D., PAI D.: DyRT: Dynamic Response Textures for Real Time Deformation Simulation With Graphics Hardware. *ACM Transactions on Graphics 21*, 3 (2002), 582–585.

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Transactions on Graphics 24*, 3 (2005), 399–407.

[KJS07] KREAVOY V., JULIUS D., SHEFFER A.: Model Composition from Interchangeable Components. In *15th Pacific Conf. on Computer Graphics and Appl.* (2007), pp. 129–138.

[KLT05] KATZ S., LEIFMAN G., TAL A.: Mesh Segmentation using Feature Point and Core Extraction. *The Visual Computer (Pacific Graph.) 21*, 8-10 (2005), 649–658.

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric Modeling in Shape Space. *ACM Transactions on Graphics 26*, 3 (2007), 1–8.

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics 22*, 3 (2003), 954–961.

[LKA06] LIEN J.-M., KEYSER J., AMATO N. M.: Simultaneous shape decomposition and skeletonization. In *Proc. of the 2006 ACM Symp. on solid and physical modeling* (2006), pp. 219–228.

[LLS*05] LEE Y., LEE S., SHAMIR A., COHEN-OR D., SEIDEL H.-P.: Mesh scissoring with minima rule and part salience. *Comp. Aided Geometric Design 22*, 5 (2005), 444–465.

[LZ07] LIU R., ZHANG H.: Mesh Segmentation via Spectral Embedding and Contour Analysis. *Computer Graphics Forum 26*, 3 (2007), 385–394.

[LZHM06] LAI Y.-K., ZHOU Q.-Y., HU S.-M., MARTIN R. R.: Feature sensitive mesh segmentation. In *Symposium on Solid and Physical Modeling* (2006), pp. 17–25.

[MPS*04] MORTARA M., PATANÈ G., SPAGNUOLO M., FALCIDIENO B., ROSSIGNAC J.: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *Proceedings of Solid Modeling and Applications* (2004), pp. 339–344.

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: modal dynamics for graphics and animation. In *Proceedings of ACM SIGGRAPH 89* (1989), pp. 215–222.

[Rus07] RUSTAMOV R. M.: Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on Geometry Processing* (2007), pp. 225–233.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing* (2007), pp. 109–116.

[Sha06] SHAMIR A.: Segmentation and Shape Extraction of 3D Boundary Meshes. In *Eurographics 2006 - State of the Art Reports* (2006), pp. 137–149.

[SP95] SCLAROFF S., PENTLAND A.: Modal Matching for Correspondence and Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 6 (1995), 545–561.

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *Visual Computer 24*, 4 (2008), 249–259.

[SY07] SCHAEFER S., YUKSEL C.: Example-based skeleton extraction. In *Symposium on Geometry Processing* (2007), pp. 153–162.

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Transactions on Graphics 26*, 3 (2007), 81.

[Wei99] WEISS Y.: Segmentation Using Eigenvectors: A Unifying View. In *Proc. ICCV* (1999), pp. 975–982.

[WK05] WU J., KOBBELT L.: Structure Recovery via Hybrid Variational Surface Approximation. *Computer Graphics Forum 24*, 3 (2005), 277–284.

[YLL*05] YAMAUCHI H., LEE S., LEE Y., OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Feature Sensitive Mesh Segmentation with Mean Shift. In *Shape Modeling and Applications* (2005), pp. 238–245.

[YLW06] YAN D.-M., LIU Y., WANG W.: Quadric Surface Extraction by Variational Shape Approximation. In *Geometric Modeling and Processing* (2006), pp. 73–86.

Appendix A: Derivation for Eq. 2

Given a function $g(\mathbf{x}) = \min_y f(\mathbf{x},\mathbf{y})$, and define $\mathbf{y}(\mathbf{x}) = \operatorname{argmin}_{\mathbf{y}} f(\mathbf{x},\mathbf{y})$ such that $g(\mathbf{x}) = f(\mathbf{x},\mathbf{y}(\mathbf{x}))$.

We observe that by definition, $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{x},\mathbf{y}(\mathbf{x})) = 0$, for all $\mathbf{x}$. This means in particular that

$$\frac{\partial}{\partial \mathbf{x}}\left[\frac{\partial f}{\partial \mathbf{y}}(\mathbf{x},y(x))\right] = \frac{\partial^2 f}{\partial \mathbf{y}\partial \mathbf{x}}(\mathbf{x},\mathbf{y}(\mathbf{x})) + \frac{\partial^2 f}{\partial \mathbf{y}^2}(\mathbf{x}) \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}(\mathbf{x}) = 0 \tag{17}$$

and we can therefore express $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}(\mathbf{x})$ as

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}(\mathbf{x}) = -\left[\frac{\partial^2 f}{\partial \mathbf{y}^2}(\mathbf{x},\mathbf{y}(\mathbf{x}))\right]^{-1} \cdot \frac{\partial^2 f}{\partial \mathbf{y}\partial \mathbf{x}}(\mathbf{x},\mathbf{y}(\mathbf{x})). \tag{18}$$

Using the chain rule, and using (17) and (18), we can expand the Hessian of g as

$$\begin{aligned}
\tfrac{\partial^2 g}{\partial \mathbf{x}^2}(\mathbf{x}) &= \tfrac{\partial^2}{\partial \mathbf{x}^2}[f(\mathbf{x},\mathbf{y}(\mathbf{x}))] \\
&= \tfrac{\partial^2 f}{\partial \mathbf{x}^2}(\mathbf{x},\mathbf{y}(\mathbf{x})) + \tfrac{\partial^2 f}{\partial \mathbf{x}\partial \mathbf{y}}(\mathbf{x},\mathbf{y}(\mathbf{x})) \cdot \tfrac{\partial \mathbf{y}}{\partial \mathbf{x}}(\mathbf{x}) \\
&= \tfrac{\partial^2 f}{\partial \mathbf{x}^2}(\mathbf{x},\mathbf{y}(\mathbf{x})) - \tfrac{\partial^2 f}{\partial \mathbf{x}\partial \mathbf{y}}(\mathbf{x},\mathbf{y}(\mathbf{x})) \cdot \left[\tfrac{\partial^2 f}{\partial \mathbf{y}^2}(\mathbf{x},\mathbf{y}(\mathbf{x}))\right]^{-1} \cdot \tfrac{\partial^2 f}{\partial \mathbf{y}\partial \mathbf{x}}(\mathbf{x},\mathbf{y}(\mathbf{x})).
\end{aligned}$$