DOI: 10.1111/cgf.12457

COMPUTER GRAPHICS *forum*
Volume 34 (2015), number 1 pp. 36–46

Example-Based Materials in Laplace–Beltrami Shape Space

Fei Zhu, Sheng Li and Guoping Wang

Department of Computer Science, Peking University, Beijing, China
{feizhu, lisheng, wgp}@pku.edu.cn

Abstract

We present a novel method for flexible and efficient simulation of example-based elastic deformation. The geometry of all input shapes is projected into a common shape space spanned by the Laplace–Beltrami eigenfunctions. The eigenfunctions are coupled to be compatible across shapes. Shape representation in the common shape space is scale-invariant and topology-independent. The limitation of previous example-based approaches is circumvented that all examples must have identical topology with the simulated object. Additionally, our method allows examples that are arbitrary in size, similar but not identical in shape with the object. We interpolate the examples via a weighted-energy minimization to find the target configuration that guides the object to desired deformation. Large deformation between examples is handled by a physically plausible energy metric. This optimization is efficient as the eigenfunctions are pre-computed and the problem dimension is small. We demonstrate the benefits of our approach with animation results and performance analysis.

Keywords: deformable simulation, example-based materials, Laplace Beltrami eigen-analysis

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism Animation

1. Introduction

The simulation of deformable materials is an essential research topic in physics-based animation because deformable objects are prevailing in animated movies, computer games and virtual reality systems. Thanks to the research progress that has been made, the deformation behaviour of many materials can be accurately reproduced with simulation. However, achieving desired deformation behaviour through tuning material parameters remains a challenging and tedious work.

Recently, Martin *et al.* [MTGG11] proposed an approach that simply uses given example poses to achieve art-directed animation of deformable materials. They employ Cauchy–Green strain as the deformation measure and construct an elastic energy that pulls current configuration towards the preferred configuration defined by examples. Their method offers an intuitive way to design materials with desired deformation behaviour in simulation at the cost of substantial computation. It involves solving an expensive nonlinear optimization problem to reconstruct a consistent geometric representation of interpolated example poses. In addition, the input examples are constrained to share the same topology with the object. In their later work [STC*12], they used an incompatible representation for the input and interpolated poses. It allows linear interpolation between elements individually and bypasses the costly reconstruction process. Performance of the method was improved, while the limitation with respect to topology remains unresolved.

Our work circumvents these limitations simultaneously, and offers additional benefits. We express shapes in a common shape space whose basis are the coupled Laplace–Beltrami eigenfunctions. In this shape space different models can be animated using the same examples (Figure 1). Computation efficiency is improved as the shape space is a reduced subspace. The contributions of our work are:

- We novelly employ the Laplace–Beltrami eigenanalysis in physics- based animation for convenient creation of art-directed deformation behaviours.
- The choice of examples is more flexible than previous methods. Examples are scale and topology independent, and can be non-identical shapes.
- Our method is efficient because the eigenfunctions are pre-computed and the most expensive run-time cost is a least-square optimization in the reduced shape space.

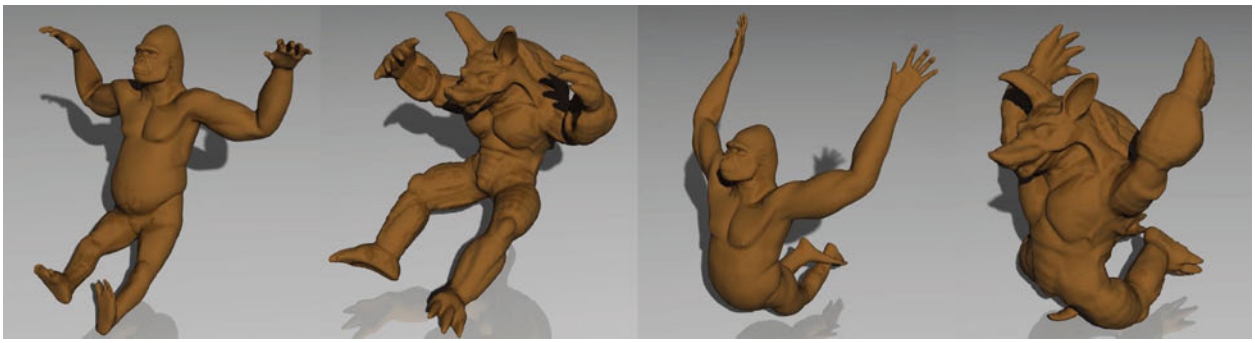

Figure 1: *A gorilla and an armadillo fall to the slippy ground with analogous reactions following common example shapes. The examples are represented by meshes different from both the gorilla and the armadillo.*

In Section 2, we review recent work that is most related to ours. Then we introduce necessary theoretical background on example-based elastic simulation and Laplace–Beltrami framework in Section 3. An overview of our method is presented in Section 4, and the details are described in Sections 5 through 7. Results are demonstrated with performance analysis in Section 8. Finally, we conclude the paper in Section 9 and Section 10 with discussions.

2. Related Work

Since the pioneering work of Terzopoulos *et al.* [TPBF87], tremendous progress has been made in accuracy, speed and robustness of the methods to simulate deformable materials. A detailed discussion on the state of the art is beyond this paper, and we suggest the survey paper of Nealen *et al.* [NMK*06] and its references. In the rest of this section, we will focus on research work most related to ours on deformable simulation control, the Laplace–Beltrami eigenanalysis and shape interpolation.

Controlling the simulation of deformable materials is essential because it is necessary in many applications. For example, an animator often wants an object to deform in a way that he desires. While realistic deformation can be achieved with physics-based simulation, controlling the deformation behaviour through manual parameter tuning is difficult if not impossible. Many methods have been proposed seeking other forms of control that are more intuitive. The prevailing kind of approaches are the space–time constraint methods. These methods control the motion of objects via minimization of an objective defined through a set of key frames. It was first proposed by Witkin *et al.* in [WK88] and improved thereafter [BP08, BdSP09, HSvTP12].

The example-based materials by Martin *et al.* [MTGG11] allow the deformation to be controlled directly by a set of example poses. Their method fits in well with the work-flow of artists. In their follow-up work [STC*12], they improved the computational efficiency and introduced the example-based plasticity model. Koyama *et al.* achieved real-time example-based elastic deformation from a geometric perspective[KTUI12]. They incorporate meshless shape matching framework and linear interpolation of the example poses. Song *et al.* [SZW*14] applied example-driven deformation in a corotational finite element framework for the purpose of fast simulation. An error–correction algorithm was proposed to address the possible errors with corotated linear strain.

The eigenvalues and eigenfunctions of the Laplace–Beltrami operator (LBO) contain intrinsic shape information and receive intensive attention in shape analysis. Reuter *et al.* [RWP05, RWP06] proposed to use the set of Laplace–Beltrami eigenvalues as a fingerprint of the shapes and they named it the 'Shape-DNA'. They presented many properties of the eigenvalues and described the numerical computation using linear/cubic FEM. As proposed in [Lev06], the potential applications of the Laplace–Beltrami eigenfunctions are wide: signal processing on surfaces, geometry processing, pose transfer, etc. Rustamov and colleagues [Rus07] presented the GPS embedding as a deformation invariant representation of non-rigid shapes, which they used for the purpose of shape classification. [Reu10] introduced a method to hierarchically segment articulated shapes into meaningful parts and to register the parts across near-isometric shapes. It exploits the isometry invariance of the Laplace–Beltrami eigenfunctions and uses the topological features for segmentation. In [OBCS*12], the authors proposed a novel representation of correspondences between shapes as linear maps between functional spaces on manifolds. The Laplace–Beltrami eigenfunctions are a good choice of basis for the functional spaces. [KBB*13] proposed an algorithm to align the eigenfunctions of multiple shapes, taking as input a set of corresponding functions such as indicator functions of stable regions on the shapes. It benefits the applications of the Laplace–Betrami eigenfunctions because the shapes are no longer restricted to be isometric/near-isometric.

Shape interpolation is an important issue in geometry processing. Numerous approaches have been proposed to avoid the artefacts of linear interpolation. One effective research trend is based on maintaining the rigid criteria of local geometrical elements or so-called as-rigid-as possible principle [ACOL00, IMH05, SA07]. These methods compute preferred interpolations by interpolating local affine transformations of the local geometrical elements. [XZWB05] proposed a non-linear gradient field interpolation method, which takes both vertex coordinates and surface orientation into account. They formulate the problem of interpolation as solving Poisson equations defined on a domain mesh. The *MeshIK* method [SZGP05] extracts per-face deformation gradients, which are interpolated between examples. The interpolated mesh is reconstructed

from the deformation gradients through a least-square optimization. [LSLCO05] introduced a rigid motion invariant mesh representation based on discrete forms defined on the mesh, and linear interpolation between the representations yields natural intermediate shapes. Winkler *et al.* [WDAH10] interpolate shapes represented by meshes in terms of edge lengths and dihedral angles. They employ a rather complicated hierarchical shape matching technique to find the mesh that best matches the interpolated edge lengths and angles. [FB11] also interpolated meshes in terms of edge lengths and dihedral angles, but derived a simpler method for computing the best matched mesh based on an elastic deformation energy of discrete shells.

3. Theoretical Background

In this section, we present an overview of the basic example-based elastic simulation framework, the LBO and its eigen spectrum. Thorough descriptions of these concepts can be found in [MTGG11] and [RWP06, Lev06, Rus07, Reu10], respectively.

3.1. Example-based elastic materials

The key idea of example-based elastic materials is to add an additional elastic potential to a basic simulator of elastic solids. The additional potential attracts the object to imitate the input examples. The equations of motion of an object discretized in space are given by

$$\mathbf{M}\ddot{\mathbf{x}} + \frac{\partial W_i}{\partial \mathbf{x}} + \frac{\partial W_p}{\partial \mathbf{x}} = \mathbf{f}_{ext}. \tag{1}$$

Here $\mathbf{x}$ and $\ddot{\mathbf{x}}$ are the positions and accelerations of the object's nodal degrees of freedom (DOFs), $\mathbf{M}$ is the mass matrix, $W_i = W(\mathbf{X}, \mathbf{x})$ represents the internal potential energy between the object's deformed configuration $\mathbf{x}$ and rest configuration $\mathbf{X}$, $W_p = W(\mathbf{X}_{target}, \mathbf{x})$ represents the potential energy between the object's deformed configuration $\mathbf{x}$ and its closest target configuration $\mathbf{X}_{target}$ on the example manifold spanned by input examples, $\mathbf{f}_{ext}$ denotes the sum of external forces due to gravity, friction and contacts.

The example manifold is a subspace of desired deformation constructed by proper interpolation of the input examples. In [MTGG11] it is constructed by linear interpolation of the elements' Green strains followed by a non-linear optimization to reconstruct a consistent configuration. In contrast, we construct the example manifold by minimizing the sum of weighted deformation energies to all the examples.

3.2. Laplace–Beltrami framework

The Laplace–Beltrami operator Δ (LBO) of any twice differentiable real-valued function on compact Riemannian manifolds can be defined as

$$\Delta f := div(grad f), \tag{2}$$

$grad$ and div are, respectively, gradient and divergence operators defined with respect to the Riemannian metric [Pet06]. The Laplace operator can be seen as the special case of the LBO with a Euclidean metric.

The spectrum of the LBO consists of eigenfunctions f_i each with a corresponding eigenvalue λ_i, which are defined as the solution to the following equation:

$$\Delta f = \lambda f. \tag{3}$$

The eigenvalues are non-negative and constitute a sequence of real-valued numbers in ascending order:

$$\lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_i \leq \ldots$$

If the manifold is closed or the Neumann boundary condition is satisfied, the first eigenvalue λ_0 will be zero and its eigenfunction f_0 becomes a constant function. The spectrum is named simple spectrum if multiplicity of all eigenvalues equals to one.

The eigenfunctions form an orthonormal basis for the space of functions defined on the manifold. The orthonormality is understood in the sense of the inner product on the manifold, which is defined as:

$$< f, g >= \int_S fg\mathrm{d}s. \tag{4}$$

We name the inner product s-inner product. Thus, normalization of the eigenfunctions requires $< f_i, f_i >= 1$.

In addition, the eigenfunctions and eigenvalues of the LBO have the following properties:

(1) Isometric shapes with simple spectrum have identical eigenfunctions up to sign. Isometric shapes with eigenvalue multiplicity greater than one have eigenfunctions up to rotation in the corresponding subspace.
(2) Similar shapes have analogous eigenvalues and eigenfunctions.
(3) Scaling an n-dimensional manifold by the factor a will scale the eigenvalues by $1/a^2$, and the s-normalized eigenfunctions by $1/a$.

Taking advantage of these properties, we are able to obtain more flexible choices of examples and better performance than previous example-based methods.

4. Method Overview

The basic idea behind our method is the construction of a common shape space that is scale invariant and topology independent. The input examples form an example manifold in this shape space, which represents the desired deformation. Through an efficient least-square optimization we find one target configuration that attracts the simulated object to the input examples.

Our method proceeds in two stages: the pre-computation stage and the run-time simulation stage. Here we outline the procedures in each stage:

- **Pre-computation:**
  (1) Compute m leading eigenfunctions/eigenvalues for the shapes and normalize the eigenfunctions (Section 5.1).
  (2) Register the examples' eigenfunctions with the object's eigenfunctions (Section 5.1).
  (3) Project the geometry of all examples onto their eigenfunctions and get corresponding coefficients as the shape descriptors (Section 5.2).
- **Run-time:**
  (1) Find the target configuration with respect to the examples and object's current configuration in the shape space (Section 6).
  (2) Reconstruct the displacement between target configuration and the object in Euclidean space, and compute the example control energy (Section 5.3).
  (3) Add the effects due to conventional deformation and external force, step the simulation.

Our method imposes no restriction on the basic elastic simulator. It can be plugged into any popular elasticity models [BW97] and time stepping techniques [MSJT08]. We use the Vega FEM library [BSS12] as the simulation framework.

All input examples are triangle surface meshes in our implementation, whereas it is not a restriction of the method, any other representations such as volumetric meshes used in [MTGG11, STC*12] works. The eigenfunction f of a manifold represented as a mesh with n vertices $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ is a vector $\boldsymbol{f} = (f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_n))^{\mathrm{T}}$. Coordinate functions $(\boldsymbol{v}_x, \boldsymbol{v}_y, \boldsymbol{v}_z)$ defined at mesh vertices can be used to describe the shape geometry, where $\boldsymbol{v}_{x\text{-}z} = (v_{x\text{-}z}(\boldsymbol{x}_1), ..., v_{x\text{-}z}(\boldsymbol{x}_n))^{\mathrm{T}}$. In this discrete setting, the s-inner product in Equation (4) is

$$< \boldsymbol{f}, \boldsymbol{g} >= \sum_{i=1}^{n} f(\boldsymbol{x}_i) g(\boldsymbol{x}_i) s(\boldsymbol{x}_i), \tag{5}$$

where $s(\boldsymbol{x}_i)$ is the surface area corresponding to $\boldsymbol{x}_i$, and the sum of $s(\boldsymbol{x}_i)$ equals the area of the surface mesh.

5. Spectral Projection and Reconstruction

In our method, shapes embedded in Euclidean space are projected into shape space spanned by the Laplace–Beltrami eigenfunctions. The displacement between the target configuration and the object is reconstructed in Euclidean space for the computation of example control energy.

5.1. Computation and registration of eigenfunctions

There exist several ways of approximating the LBO and its eigenfunctions on discrete representations of manifolds (see [RBG*09] for a comparison). We choose the linear/cubic FEM approximation [RWP06, Reu10] because of its high accuracy. The first constant eigenfunction is excluded from the computed eigenfunctions since it encodes only rotation and rigid translation.

Figure 2: *The eigenfunctions of a deformed armadillo are registered with respect to the eigenfunctions of a gorilla. After the registration, the eigenfunctions between the shapes are consistent. Hot and cold colours represent positive and negative values, respectively.*

The eigenfunctions of different shapes need to be aligned before they can be used as the common (approximate) basis. For multiple isometric/near-isometric shapes with simple spectrum, the eigenfunctions are consistent up to sign which can be aligned with simple *sign-flipping* operations. However this is not guaranteed for general shapes. In the case of non-trivial eigenvalue multiplicity, the alignment of eigenfunctions involves rotation. Besides, the order of the eigenfunctions across shapes is not consistent due to numerical instabilities. The eigenfunctions may vary significantly if the shapes deviate beyond certain degree. We employ the method in [KBB*13] to couple the eigenfunctions. Given a set of corresponding information between the shapes, the algorithm rotates and reflects the eigenfunctions such that they're best aligned in the least-squares sense. We input the correspondence of some regions on the shapes and register the eigenfunctions of all examples according to the object's eigenfunctions (Figure 2). The corresponding regions can be easily acquired using the algorithm described in [LBB11] or manual marking. The registration is convenient and needs to be conducted only once, because the adjusted eigenfunctions can be saved to disk for later simulation.

5.2. Projection onto eigenfunctions

We project the examples and the object onto their Laplace–Beltrami eigenfunctions. The projection is done through s-inner product of shape's geometry and its Laplace–Beltrami eigenfunctions. We know that if an n-dimensional manifold is scaled by the factor a, area of the manifold will be scaled by a^2, the eigenvalues by $1/a^2$, and the s-normalized eigenfunctions by $1/a$. Thus, we scale each eigenfunction $\boldsymbol{f}_i$ with λ_i so that size information is removed from the inner-product of shape geometry with the eigenfunctions. The set of eigenfunctions used for the projection are:

$$\{\lambda_1 \boldsymbol{f}_1, \lambda_2 \boldsymbol{f}_2, \ldots, \lambda_m \boldsymbol{f}_m\}.$$

The shape geometry is represented as coordinate functions of mesh vertices $\boldsymbol{v}(\boldsymbol{v}_x, \boldsymbol{v}_y, \boldsymbol{v}_z)$. Hence projection of the shape geometry

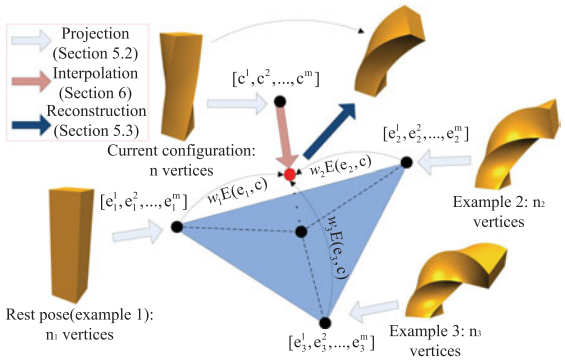

Figure 3: *Overview of run-time operations. Three poses of the elastic cuboid are given as examples here, each with different number of vertices. The examples, as well as the object are projected onto their first m Laplace–Beltrami eigenfunctions. Each projected shape is represented as an m-dimensional point. The target configuration (red point) is minimizer of weighted energies to all the examples.*

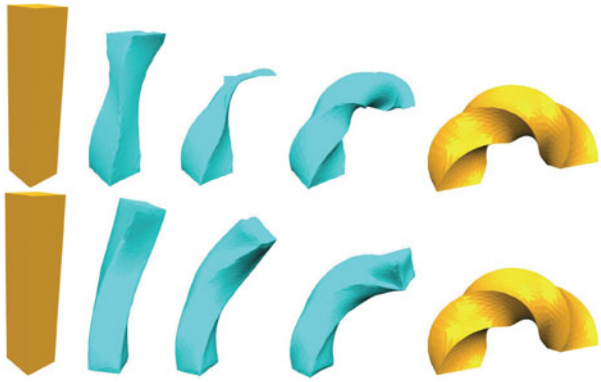

Figure 4: *Comparison of linear interpolation (top) and our approach (bottom) between two severely deformed shapes (yellow). The interpolated shapes (cyan) are shown for interpolation weights of $w_1 = 0.25$, $w_1 = 0.5$, $w_1 = 0.75$.*

can be computed as:

$$(< \boldsymbol{v}_x, \lambda_i \boldsymbol{f}_i >, < \boldsymbol{v}_y, \lambda_i \boldsymbol{f}_i >, < \boldsymbol{v}_z, \lambda_i \boldsymbol{f}_i >) \; (1 \leq i \leq m).$$

It results in an $m \times 3$ matrix with columns corresponding to the projection of the three coordinate components. The matrix can also be considered as a generalized m-dimensional vector in the m-dimensional shape space (Figure 3).

5.3. Reconstruction from eigenfunctions

For every point in the Laplace–Beltrami shape space, we can reconstruct its representation in Euclidean space with the eigenfunctions. The size information is embedded back by scaling the eigenfunctions in correspondence to the projection process (Section 5.2). The eigenfunctions used for the reconstruction are:

$$\left\{ \frac{1}{\lambda_1} \boldsymbol{f}_1, \frac{1}{\lambda_2} \boldsymbol{f}_2, \ldots, \frac{1}{\lambda_m} \boldsymbol{f}_m \right\}.$$

We compute the displacement between the target configuration and object's current configuration in the m-dimensional shape space. Suppose it is $\boldsymbol{d}(d^1, d^2, ..., d^m)$, then the displacement in Euclidean space is reconstructed using the eigenfunctions of the object as $\sum_{i=1}^{m} d^i \frac{1}{\lambda_i} \boldsymbol{f}_i$.

The reconstructed displacement is defined on the surface as we represent shapes with surface meshes. We then distribute displacements of the surface vertices to vertices of the object's volumetric mesh using a compact-supported decreasing kernel. In our implementation, we use a linear kernel. A volumetric mesh vertex may be in the kernel range of several surface vertices. Contributions from all these surface vertices are averaged. Note that this 'displacement distribution' operation can be bypassed if volumetric meshes are used for eigenfunction computation.

From the displacements of the volumetric mesh vertices, we construct an elastic energy that pulls the object to the target configuration. Any elasticity model can be used to compute the energy, and a simple linear elastic model suffices in our experiments. The material stiffness is scaled by a scaling factor α to addjust the strength of examples.

6. Interpolation of Examples

Suppose that we are given k examples, represented as m-dimensional points $\boldsymbol{e}_i$ $(1 \leq i \leq k)$ in the shape space spanned by the eigenfunctions. We interpolate the examples to get one target configuration that guides the deformation of the object. Linear interpolation leads to artefacts when the deformation between the examples is severe (Figure 4, top). We incorporate an efficient weighted-energy interpolation approach, so that intermediate configurations between the examples are naturally deformed (Figure 4, bottom).

The target configuration $\boldsymbol{t}$ is defined as the one that minimizes the sum of weighted deformation energies to all the examples (Figure 3):

$$\min_{\boldsymbol{t}} \sum_{i=1}^{k} w_i \mathbf{E}(\boldsymbol{t}, \boldsymbol{e}_i), \tag{6}$$

where $\mathbf{E}(\cdot, \cdot)$ is some non-linear deformation energy between two shapes, w_i $(1 \leq i \leq k)$ measures the guiding strength of the examples and $\sum_{i=1}^{k} w_i = 1$.

We dynamically determine the guiding strength of each example according to its closeness with the object's configuration $\boldsymbol{c}$ in each time step:

$$\min_{w_i} \frac{1}{2} \| \sum_{i=1}^{k} w_i \boldsymbol{e}_i - \boldsymbol{c} \|_F^2 \tag{7}$$

s.t.

$$0 \leq w_i \leq 1, \quad i = 1 \ldots k$$

$$\sum_{i=1}^{k} w_i = 1,$$

where $\| \cdot \|_F$ is the Frobenius norm.

Equations (6) and (7) can be formulated as a single optimization problem by enforcing Equation (7) as constraints of Equation (6):

$$\min_{\boldsymbol{t}, w_i} \sum_{i=1}^{k} w_i \mathbf{E}(\boldsymbol{t}, \boldsymbol{e}_i) \tag{8}$$

s.t.

$$\left(\sum_{i=1}^{k} w_i \boldsymbol{e}_i - \boldsymbol{c}\right) \cdot \boldsymbol{e}_j = 0, \quad j = 1 \ldots k,$$

$$0 \leq w_i \leq 1, \quad i = 1 \ldots k,$$

$$\sum_{i=1}^{k} w_i = 1.$$

[RW09, CPSS10] employed similar weighted-energy interpolation strategy to get intermediate shapes between given shapes. Their approaches only apply to shapes that have identical topology and the unknowns are the vertex positions, whereas we represent shapes of different topologies in the common Laplace–Beltrami shape space and the number of unknowns is reduced to the number of eigenfunctions.

$\mathbf{E}(\cdot, \cdot)$ can be arbitrary non-linear deformation energy between two shapes that is physically plausible. We choose the energy in [FB11] since all the shapes are represented as surface meshes in our implementation. The energy is composed of three terms: stretching term $\mathbf{E}_s$, bending term $\mathbf{E}_b$ and volume preservation term $\mathbf{E}_v$. They each measures the squared deviations of edge lengths, dihedral angles and the volume enclosed by the shapes. All the three terms can be represented by vertex positions of the meshes, and in turn by the coefficient vectors in the Laplace–Beltrami shape space. For shape $\boldsymbol{e}$ that does not have compatible topology with the object, we can reconstruct a compatible representation with the object's eigenfunctions:

$$\boldsymbol{x} = \boldsymbol{x}_{rest} + \sum_{i=1}^{m} \left(\boldsymbol{e}^i - \boldsymbol{c}^i_{rest}\right) \frac{1}{\lambda_i} \boldsymbol{f}_i. \tag{9}$$

$\boldsymbol{x}_{rest}$ is vertex positions of the object's initial configuration, $\boldsymbol{c}_{rest}$ is the corresponding coefficient vector. Therefore energy between $\boldsymbol{e}$ and $\boldsymbol{c}$ can be represented by their corresponding coefficient vectors.

The optimization objective is quadratic with respect to the coefficient vector when $\mathbf{E}(\cdot, \cdot)$ is the deformation energy in [FB11]. It leads to a simple linear system to solve. The number of unknowns is $3 \times m + k$, which is small compared to the number of vertices. We use the Conjugate Gradient solver in ALGLIB (http://www.alglib.net/) to solve this problem. Current configuration $\boldsymbol{c}$ is used as the initial guess, and the solver typically converges in less than 10 iterations.

7. Extension to Local Examples

We have thus far assumed that deformation of the examples influences the object globally. However, extension of our method to employing *local* examples that only influence local regions of the object is straightforward. We partition the entire domain of the object into separate regions, each influenced by the corresponding region on examples independently. The corresponding regions on different shapes need not to have identical resolution. For each local region on the object, we project only the part of geometry that is in this region onto the entire object's Laplace–Beltrami eigenfunctions. This is achieved by setting entries corresponding to vertices outside the region in the vector representing shape geometry to zero. The example manifold for this region is constructed by projecting the region's counterparts on the example poses onto corresponding eigenfunctions. After reconstructing the displacement vector between current configuration and target configuration for each region (Section 5.3), we extract entries of vertices in the region and proceed with rest steps of the method. Each local region is handled independently, thus our method allows for different local examples combined together to yield complex behaviours.

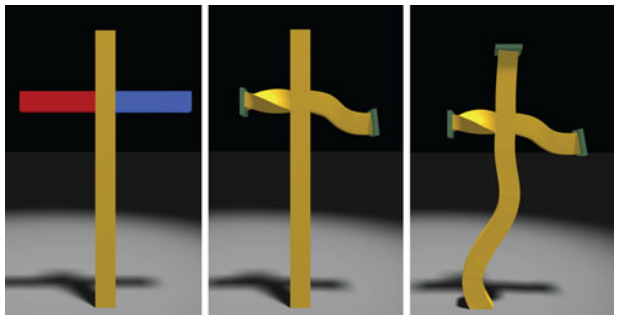

Figure 5: *Eigenfunctions of the three local regions on the cross shape are computed, respectively (left). Compression on the two horizonal regions (middle) followed by compression on the vertical region (right) leads to different fine-scale deformation styles on each region.*

To resolve fine-scale deformation of local regions, a slight modification is incorporated. We pre-compute Laplace–Beltrami eigenfunctions for each local region and project the region's geometry onto the local-support eigenfunctions instead of onto eigenfunctions of the entire shape as mentioned before. Similar multi-domain strategy has been exploited in [KJ11, BZ11] to capture local deformation details without compensating the benefits of reduced simulation. Different from their method, our multi-domain strategy is used only to compute guiding effect exerted by the examples and the simulation is still conducted on the entire domain. Thus, there's no need to handle inter-domain cracking artefacts. Figure 5 is one example illustrating the effect of this strategy.

8. Results

In this section, we demonstrate that we can create results visually very similar to previous approaches, but with more flexible choices of examples. We also conduct performance comparison and scaling analysis to show the efficiency of our method.

Figure 6 shows a simple example of an elastic bar deforming under gravity. We believe we have successfully reproduced the deformation behaviour presented by Martin *et al.* [MTGG11]. It's worth

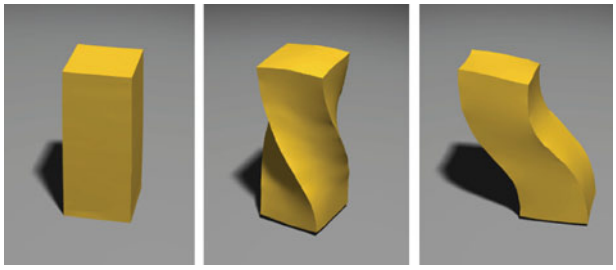

Figure 6: *An elastic bar deforms under gravity using no example, a twisted example and an S-shaped example, respectively.*

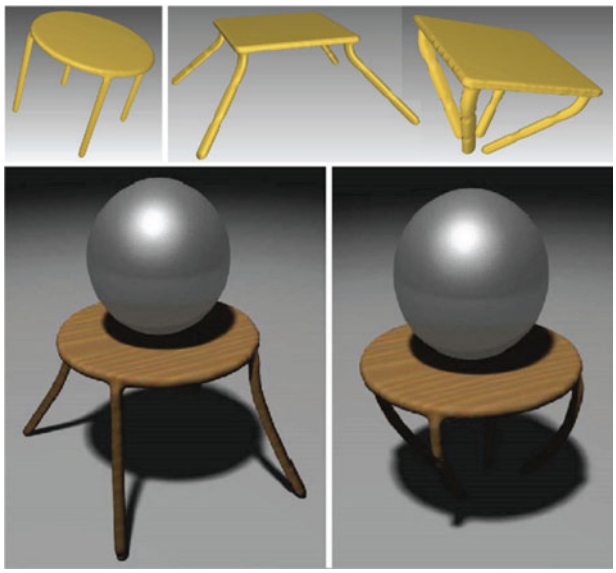

Figure 7: *A round stool bends its legs while compressed by a steel sphere (bottom). Square stools are provided as examples (top).*

noting that meshes with different resolutions are used to compute the eigenfunctions for the object and the examples.

As demonstrated in the accompanying video, our method allows guiding the simulation of an elastic cuboid bar with cylindroid bars. The deformation behaviours produced using cylindroid bar examples are similar to those in Figure 6. This further illustrates the flexibility of our approach in choosing examples compared with previous approaches.

Figure 7 is one more example that demonstrates this flexibility of our method. The legs of the round stool deform exactly like the square stools provided as examples.

As seen in Figure 8, a gummy bear equipped with expressive examples can perform peculiar reactions to user interactions. Figure 9 depicts the effect of local examples on the teddy bear to generate more interesting deformation behaviours. The left and right sides of the teddy bear can be manipulated independently because each side is influenced only by corresponding local region from the example pose.

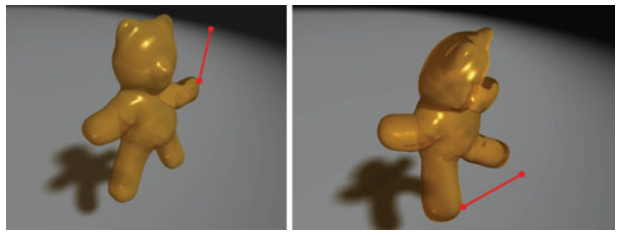

Figure 8: *A gummy bear puts up both its arms (left) and bends down its head (right) in response to different user interactions. The red lines depict the user interactions.*

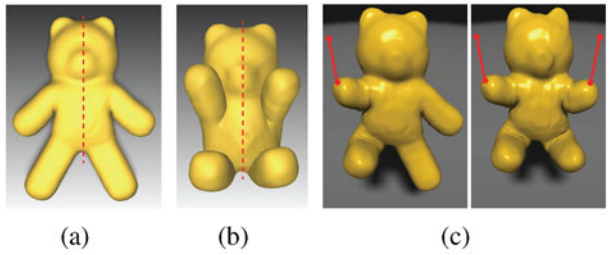

Figure 9: *The teddy bear (a) is divided into two separate regions, each guided by corresponding region from one example pose (b). The teddy bear's left and right parts deform independently in response to user interactions (c).*

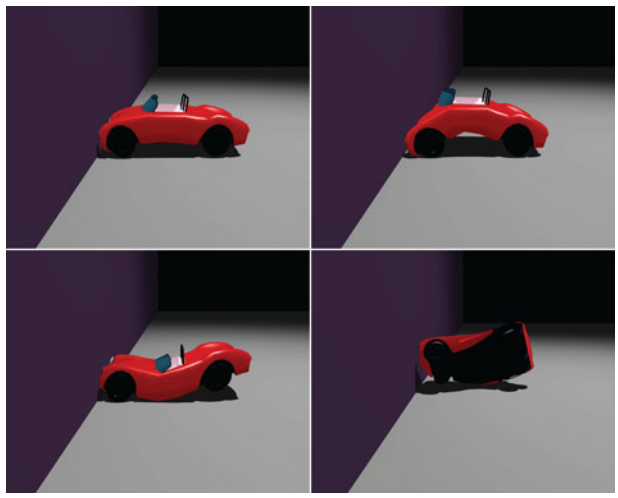

Figure 10: *A sports car hits the wall and reacts in diverse ways with no example (upper left), an arc-shaped example (upper right), a wave-shaped example (lower left) and a twisted example (lower right).*

We emulate a toy sports car hitting the wall following different examples in Figure 10. As the shape of a car can be described by its body, we employ meshes of the car body for eigenfunction computation and example design.

The examples to guide the plane model in Figure 11 vary from the plane itself to a bird, and even a fish. Different examples are

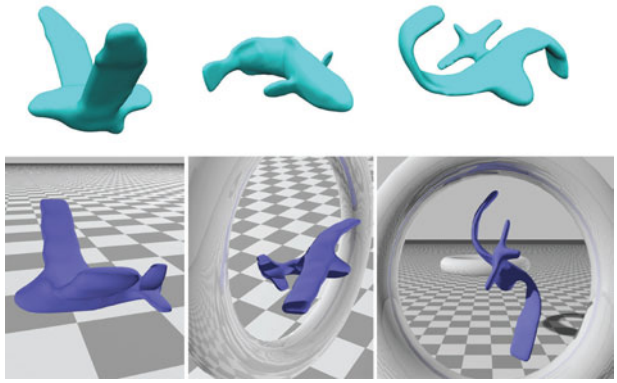

Figure 11: *A toy plane performs a series of artistic moves (bottom) following different example shapes (top).*

Table 1: *Summary of all results presented in the paper. The columns indicate the number of DOFs, vertex number of the mesh to compute the object's eigenfunctions, the number of eigenfunctions, average time (in milliseconds) for example interpolation and total time for one simulation step.*

Model	#DOFs	#V(obj)	#Eigen	t_{interp}	t_{tot}
Gorilla	4659	2225	5	227.3	247.6
Armadillo	7814	3170	5	357.4	396.5
Bar (cuboid ex.)	981	682	14	51.2	55.5
Bar (cylindroid ex.)	981	5811	15	266.8	271.6
Stool	1608	11 447	3	111.7	121.4
Teddy arms up	2571	11 273	6	86.7	105.2
Teddy head down	1773	11 273	6	70.3	83.5
Teddy local ex.	1773	11 273	6	302.8	313.7
Cross	4683	682	14	564.1	592.4
Car	2181	9441	4	207.7	221.6
Plane	7701	4728	5	340.6	424.9
E	6501	4199	4	186.9	232.7
G	3129	5633	4	335.0	351.5
2	2364	4467	4	318.6	328.3
0	2445	4430	4	301.7	314.4
1	2421	2995	4	177.8	189.9
4	4887	3365	4	272.9	303.6

Table 2: *Performance scaling (in milliseconds) for multiple examples (top row) illustrated in the 'Cuboid Twist' animation for one single iteration.*

#ex.	1 ex.	2 ex.	3 ex.	4 ex.	8 ex.	12 ex.
t (ms)	8.5	15.9	23.2	30.5	59.8	85.1

Table 3: *Performance scaling (in milliseconds) for different resolutions of the mesh for eigenfunction computation (top row) illustrated in the 'Cuboid Twist' animation for one single iteration.*

#vert	1000	2000	4000	6000	8000	10 000
t (ms)	21.2	45.4	112.6	155.1	213.5	297.0

activated during the process of the plane's move using the position of the plane's centre as a trigger. The plane flaps its wings like a bird while taking off, bends the body to turn the direction following a fish example and rotates through a torus with an impressive pose.

In Figure 12, we design various deformation behaviours for the letter-shaped soft bodies and make them react to impacts following the augmented deformation styles. The 'E', '2' and '1' are crushed by another object with their bottom parts fixed on the ground, while the other letters fall and hit the ground under gravity. The deformation depicted in this figure, for example, 'G' swinging its upper body and '0' twisting, cannot be easily produced merely by physics. With our example-based approach, we can efficiently create these cartoon style behaviours.

8.1. Performance

Timing information for all results presented in this paper is summarized in Table 1. The simulations are run on a single core of an Intel Core i5, 2.8GHz (Santa Clara, CA, USA), with no particular code optimization. Time step size is uniformly set to 0.005 s for all simulations. As can be seen in the table, the run-time cost is dominated by the interpolation of examples to find the target configuration. Due to the decoupling of physical simulation and shape interpolation in the Laplace–Beltrami shape space, the cost of shape interpolation is constant with respect to the simulation DOFs. It only depends on the number of eigenfunctions, number of examples and resolution

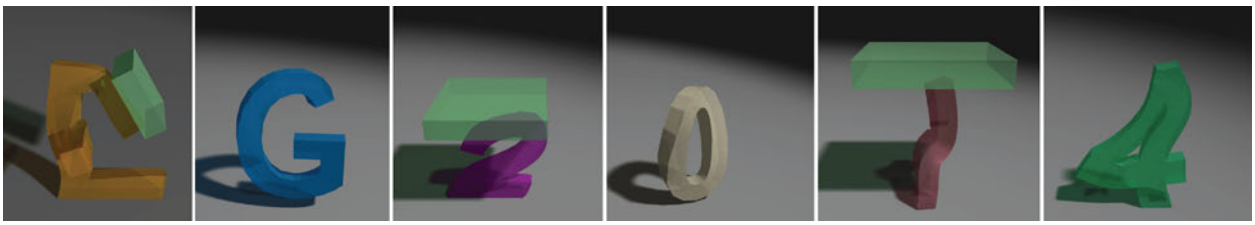

Figure 12: *Letter-shaped soft bodies perform designed deformation behaviours in response to different impact events.*

of the mesh to compute the object's eigenfunctions. Therefore we can simulate with much more DOFs than Martin *et al.*'s approach [MTGG11] and still manage to lead in efficiency. Their method typically takes seconds for simulation with the number of DOFs around 1000. In contrast we can achieve several frames per second with times more DOFs than theirs. Estimating the time for synthesizing 1 s of simulated motions from the timings in the table and the size of time step, we find that our method is roughly as efficient as Schumacher *et al.*'s method [STC*12] at the same scale of simulation complexity and even faster in some cases. The 'Cuboid Twisting' for instance, our approach costs 55.5 ms, Martin *et al.*'s needs 528∼3064 ms and Schumacher *et al.*'s claims to be 14.8 times faster than Martin *et al.*'s which is approximately 35.7∼207 ms. The number of DOFs is approximately identical for the three methods.

We also measure the timings of the 'Cuboid Twist' animation using different number of examples for a quantitative comparison with [MTGG11]. The results in Table 2 indicate that the time spent on solving for the target configuration scales linearly with the number of examples. This is reasonable as our weighted-energy optimization evaluates deformation energy from the target configuration to every example. Comparing the timings with those in table 2 of [MTGG11], we can see that our method outperforms theirs for all the listed number of examples.

This is due to the fact that the dimension of our optimization problem is much smaller. Generally less than 10 examples would suffice for the creation of impressive animations and introducing too many examples can create ambiguities about the desired deformation. Thus our method is evidently more efficient for practical use despite that they achieve better scaling than ours.

Table 3 lists timing information of the same animation using meshes with different resolutions to compute eigenfunctions of the simulated object. The cost increases with the complexity of the mesh because evaluation of the objective function at each optimization iteration involves the number of edges and faces (Section 6). The performance scaling is acceptable and the mesh need not to be very dense in practice. We manage to achieve beneficial efficiency by restricting the maximum complexity of the meshes to about 10 000 vertices.

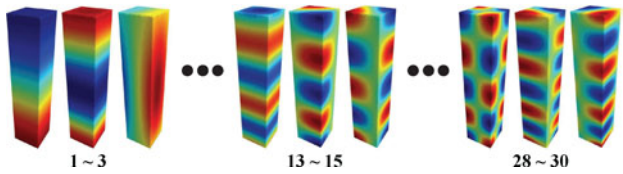

Figure 13: *Visualization of the first 30 Laplace–Beltrami eigenfunctions of a cuboid bar. Eigenfunctions corresponding to ascending eigenvalues capture deformation from coarse to fine scale.*

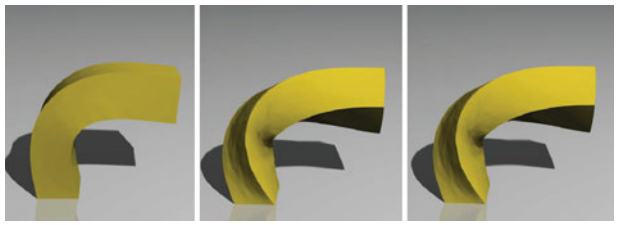

Figure 14: *Elastic bar guided by example with coupled twisting and bending, using different number of eigenfunctions. Two eigenfunctions (left) fail to capture the twisting, while 14 eigenfunctions (middle) are sufficient to capture both bending and twisting. Increasing the number of eigenfunctions to 30 generates similar result (right).*

Figure 15: *A gorilla equipped with different number of eigenfunctions to mimic the deformation of the example (left). Two eigenfunctions (middle) capture only the deformation of the arms, while three eigenfunctions (right) reproduce the deformation of arms and legs.*

9. Discussion

9.1. Number of eigenfunctions

The sequence of Laplace–Beltrami eigenfunctions corresponding to ascending eigenvalues describe the shapes' features from coarse to fine scale (Figure 13). Using the truncated set of leading eigenfunctions as the basis of deformation space inevitably loses some high-frequency details. However we consider this as an acceptable trade-off since the choice of examples is more flexible and shape interpolation is more efficient in the reduced shape space. The minimum size of the set of eigenfunctions depends on the actual shapes and the deformation of the examples. The eigenfunctions must at least be able to capture the major deformation of the examples. Generally, a small number of eigenfunctions are adequate. For example, in Figure 14 the coupled twisting and bending of the example can be captured by only 14 eigenfunctions and 30 eigenfunctions are not necessary as the results are visually identical. Figure 15 is one more example where two eigenfunctions only reproduce the deformation of the arms and three eigenfunctions are needed in order to fully capture the deformation of the four limbs. Visualization of the gorilla's eigenfunctions is shown in Figure 2. It is in fact quite simple to choose the proper number of eigenfunctions using the visualization of eigenfunctions for reference. In the limit case where full set of eigenfunctions are used, the ability of using examples of arbitrary sizes and topology still makes our method attractive.

9.2. Comparison to deformation transfer

One alternative approach that can achieve the same flexibility as ours is to first transfer the deformation of the examples to the object using existing deformation transfer techniques [SP04, BVGP09, ZXTD10] and then interpolate the deformed object poses as in [MTGG11, STC*12] to find the target configuration. We do not

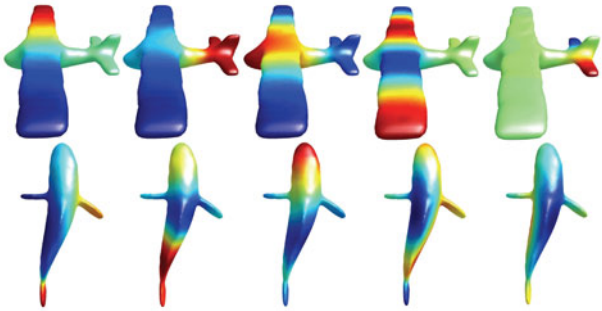

Figure 16: *The first five eigenfunctions of a plane and a fish after the registration still show noticeable difference.*

explicitly transfer deformation between meshes in Euclidean space. Instead we find the target configuration directly in the Laplace–Beltrami shape space. For explicit deformation transfer the problem dimension of finding the target configuration is the number of elements on the object. In contrast, the problem dimension of ours is the number of eigenfunctions which is much smaller. Thus our approach also outperforms in computation efficiency.

9.3. Limitations

The flexibility of our method in the choice of examples has been demonstrated with compelling results, but much work remains to be done. Our method relies on the eigenfunctions of input shapes to be consistent, which is stringent for two notably different shapes even if after the registration. For example, eigenfunctions of the plane and fish in Figure 16 show noticeable difference although after the registration operation. We managed to guide the bending of the plane with the fish using the first three eigenfunctions (Figure 11), but it is impractical for deformation of higher frequency as the eigenfunctions start to vary significantly from the fourth column. From an animator's perspective, there exists some intuitive correspondence between a plane and a fish. Thus it would be interesting to explore the intuitive shape correspondence that is beyond the capability of the Laplace–Beltrami eigenfunctions.

10. Conclusion

We have made the simulation of example-based materials more flexible and efficient by representing shapes in the Laplace–Beltrami shape space. Our method extends the range of input examples and reduces the computation cost. It is the first example-based method to achieve the two gains simultaneously.

Acknowledgements

We would like to thank the anonymous reviewers for the invaluable comments. We are also grateful to Tianxiang Zhang, Sheng Yang and Meihan Wu for their generous help with the demo production.

This research was supported by Grant No. 2010CB328002 from The National Basic Research Program of China (973 Program), Grant No. 61232014, 61121002, 61173080 from National Natural Science Foundation of China and Grant No. 2013BAK03B07 from The National Key Technology Research and Development Program of China.

References

[ACOL00] Alexa M., Cohen-Or D., Levin D.: As-rigid-as-possible shape interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 157–164.

[BdSP09] Barbič J., da Silva M., Popović J.: Deformable object animation using reduced optimal control. In *Proceedings of SIGGRAPH, ACM Transactions on Graphics 28*, 3 (2009), pp. 53:1–53:9.

[BP08] Barbič J., Popović J.: Real-time control of physically based simulations using gentle forces. In *ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 163:1–163:10.

[BSS12] Barbič J., Sin F. S., Schroeder D.: Vega FEM Library, 2012. http://www.jernejbarbic.com/vega. Accessed 30 May 2013.

[BVGP09] Baran I., Vlasic D., Grinspun E., Popović, J.: Semantic deformation transfer. *ACM Transactions on Graphics 28*, 3 (July 2009), 36:1–36:6.

[BW97] Bonet J., Wood R. D.: *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, New York, 1997.

[BZ11] Barbič J., Zhao Y.: Real-time large-deformation substructuring. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 91:1–91:8.

[CPSS10] Chao I., Pinkall U., Sanan P., Schröder P.: A simple geometric model for elastic deformations. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 38:1–38:6.

[FB11] Fr Aïlich S., Botsch M.: Example-driven deformations based on discrete shells. *Computer Graphics Forum 30*, 8 (2011), 2246–2257.

[HSvTP12] Hildebrandt K., Schulz, C., von Tycowicz C., Polthier K.: Interactive spacetime control of deformable objects. *ACM Transactiosn on Graphics 31*, 4 (July 2012), 71:1–71:8.

[IMH05] Igarashi T., Moscovich T., Hughes J. F.: As-rigid-as-possible shape manipulation. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 1134–1141.

[KBB*13] Kovnatsky A., Bronstein M. M., Bronstein A. M., Glashoff K., Kimmel R.: Coupled quasi-harmonic bases. *Computer Graphics Forum 32*, 2pt4 (2013), 439–448.

[KJ11] Kim T., James D. L.: Physics-based character skinning using multi-domain subspace deformations. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 63–72.

[KTUI12] Koyama Y., Takayama K., Umetani N., Igarashi T.: Real-time example-based elastic deformation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2012), SCA '12, Eurographics Association, pp. 19–24.

[LBB11] Litman R., Bronstein A. M., Bronstein M. M.: Diffusion-geometric maximally stable component detection in deformable shapes. *Computers & Graphics 35*, 3 (2011), 549–560. [Shape Modeling International (SMI) Conference 2011.]

[Lev06] Levy B.: Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006* (Washington, DC, USA, 2006), SMI '06, IEEE Computer Society, pp. 13–.

[LSLCO05] Lipman Y., Sorkine O., Levin D., Cohen-Or D.: Linear rotation-invariant coordinates for meshes. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 479–487.

[MSJT08] Müller M., Stam J., James D., Thürey N.: Real time physics: class notes. In *ACM SIGGRAPH 2008 Classes* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 88:1–88:90.

[MTGG11] Martin S., Thomaszewski B., Grinspun E., Gross M.: Example-based elastic materials. *ACM Transactions on Graphics 30*, 4 (July 2011), 72:1–72:8.

[NMK*06] Nealen A., Mller M., Keiser R., Boxerman E., Carlson M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836.

[OBCS*12] Ovsjanikov M., Ben-Chen M., Solomon J., Butscher A., Guibas L.: Functional maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics 31*, 4 (July 2012), 30:1–30:11.

[Pet06] Petersen P.: *Riemannian Geometry. Graduate Texts in Mathematics*. Springer, New York, 2006.

[RBG*09] Reuter M., Biasotti S., Giorgi D., Patanè G., Spagnuolo M.: Technical section: Discrete laplace-beltrami operators for shape analysis and segmentation. *Comput. Graph. 33*, 3 (June 2009), 381–390.

[Reu10] Reuter M.: Hierarchical shape segmentation and registration via topological features of Laplace-Beltrami eigenfunctions. *International Journal on Computor Vision 89*, 2–3 (Sept. 2010), 287–308.

[Rus07] Rustamov R. M.: Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), SGP '07, Eurographics Association, pp. 225–233.

[RW09] Rumpf M., Wirth B.: A nonlinear elastic shape averaging approach. *SIAM Journal on Image Science 2*, 3 (July 2009), 800–833.

[RWP05] Reuter M., Wolter F.-E., Peinecke N.: Laplace-spectra as fingerprints for shape matching. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2005), SPM '05, ACM, pp. 101–106.

[RWP06] Reuter M., Wolter F.-E., Peinecke N.: Laplace-Beltrami spectra as 'shape-dna' of surfaces and solids. *Computer Aided Design 38*, 4 (April 2006), 342–366.

[SA07] Sorkine O., Alexa M.: As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), SGP '07, Eurographics Association, pp. 109–116.

[SP04] Sumner R. W., Popović J.: Deformation transfer for triangle meshes. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, ACM, pp. 399–405.

[STC*12] Schumacher C., Thomaszewski B., Coros S., Martin S., Sumner R., Gross M.: Efficient simulation of example-based materials. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2012), SCA '12, Eurographics Association, pp. 1–8.

[SZGP05] Sumner R. W., Zwicker M., Gotsman C., Popović J.: Mesh-based inverse kinematics. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 488–495.

[SZW*14] Song C., Zhang H., Wang X., Han J., Wang H.: Fast corotational simulation for example-driven deformation. *Computers & Graphics* 40 (2014), 49–57.

[TPBF87] Terzopoulos D., Platt J., Barr A., Fleischer K.: Elastically deformable models. *SIGGRAPH Computer Graphics 21*, 4 (Aug. 1987), 205–214.

[WDAH10] Winkler T., Drieseberg J., Alexa M., Hormann K.: Multi-scale geometry interpolation. *Computer Graphics Forum 29*, 2 (2010), 309–318.

[WK88] Witkin A., Kass M.: Spacetime constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), SIGGRAPH '88, ACM, pp. 159–168.

[XZWB05] Xu D., Zhang H., Wang Q., Bao H.: Poisson shape interpolation. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2005), SPM '05, ACM, pp. 267–274.

[ZXTD10] Zhou K., Xu W., Tong Y., Desbrun M.: Deformation transfer to multi-component objects. *Computer Graphics Forum 29*, 2 (2010), 319–325.