Multimed Tools Appl (2013) 62:581–599
DOI 10.1007/s11042-011-0860-8

Sketch-based design for green geometry and image deformation

Bin Sheng · Weiliang Meng · Hanqiu Sun · Enhua Wu

Published online: 25 August 2011

Abstract User interfaces have traditionally followed the WIMP (window, icon, menu, pointer) paradigm. Though functional and powerful, they are usually cumbersome for a novice user to design a complex model, requiring considerable expertise and effort. This paper presents a system for designing geometric models and image deformation with sketching curves, with the use of Green coordinates. In 3D modeling, the user first creates a 3D model by using a sketching interface, where a given 2D curve is interpreted as the projection of the 3D curve. The user can add, remove, and deform these control curves easily, as if working with a 2D line drawing. For a given set of curves, the system automatically identifies the topology and face embedding by applying graph rotation system. Green coordinates are then used to deform the generated models with detail-preserving property. Also, we have developed a sketch-based image-editing interface to deform image regions using Green coordinates.

B. Sheng (✉)
Department of Computer Science and Engineering, Shanghai Jiao Tong University,
Shanghai, China
e-mail: shengbin@cs.sjtu.edu.cn

B. Sheng
Department of Computer Science and Engineering, The Chinese University of Hong Kong,
Shatin, Hong Kong, China

W. Meng
Institute of Software, Chinese Academy of Sciences,
LIAMA-NLPR, CAS Institute of Automation, Beijing, China

H. Sun
The Chinese University of Hong Kong, Shatin, Hong Kong, China

E. Wu
Institute of Software, Chinese Academy of Sciences, Beijing, China

E. Wu
University of Macau, Macau, China

Hardware-assisted schemes are provided for both control shape deformation and the subsequent surface optimization, the experimental results demonstrate that 3D/2D deformations can be achieved in realtime.

Keywords Sketch-based deformation · Green coordinates · Optimization · Laplace's equation

1 Introduction

While recent decades have seen significant progress in CAD software, current state of the art still appears insufficient when it comes to the styling design of products. This is evidenced by the fact that a significant portion of early design activities such as concept development and style generation occurs almost exclusively in 2D environments—be it the traditional pen-and-paper environment or its digital equivalents. While part of this bias toward 2D tools in the early design stages comes from the undeniable convenience and familiarity of such media, we believe the lack of suitable software and interaction techniques to support 3D/2D styling design has a significant role in the current bias.

A group of research tools [15, 16, 22, 28, 38] as well as in-game character editors [2] are built around intuitive modeling metaphors. However, some of these tools lack a high-level control structure, making it difficult to iteratively refine the design, or re-use existing designs. Nowadays, most designers still prefer to express their creative ideas through 2D sketches. 2D user interface has problems in interpreting 2D curves and surfaces into 3D entities. One innovative work was done by [16], where the system provides a planar canvas first and the user starts with a planar closed curve.

In this paper, we propose a sketch-based modeling system for 3D/2D styling design. Our system lets users create and edit 3D geometry through direct sketching. A distinguishing feature of our system is that we tailored it toward the rapid and intuitive design of styling deformation with detail-preservation features such as free-form curves and surfaces–which is often a tedious and complicated task using conventional software. Our sketch-based deformation system adopts Green coordinates [24] which leads to space deformations with a shape-preserving property, inducing conformal mappings (2D) and quasi-conformal mappings (3D). In our 3D sketch-based modeling system, a given 2D curve is interpreted as the projection of 3D curve with the minimum curvature. We also developed a data structure, called *Bidirectional Patch Table* (BPT), to support topological computation and efficiently identify faces on the mesh. Another issue worth considering is how to generate smooth surfaces. We generate the 3D surface shape in the form of a 3D parametric surface, which is obtained by transforming the 2D plane into 3D surface and automatically interpolating the mesh vertices of the 2D triangle mesh into 3D. After the 3D model is generated, we can construct a bounding cage for the model, which is then embed into the cage space where Green coordinates are constructed.

Our sketch-based deformation is also applied to 2D warping and deformation. The user can interactively sketch the cage of the region-of-interest (ROI), and the

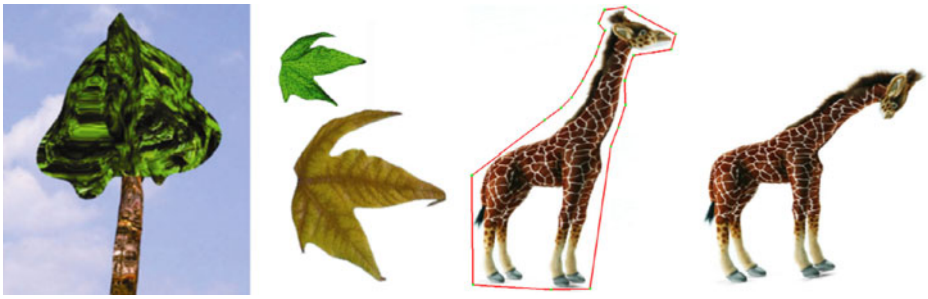


Fig. 1 Geometry modeling and image deformation by using our sketch-based design and Green coordinates

deformation effect is achieved by simply changing the position of cage vertices by interpolating the Green coordinates. Therefore, our sketch-based image deformation engages non-linear interpolation, a basis for preserving shapes and details inherently (see Fig. 1). Our sketch-based deformation has been implemented with realtime feedback, due to the acceleration of graphics hardware (GPU). Such real-time, intuition-based user interaction enables easy image animation by allowing 2D motions of cage vertices, as demonstrated by our experimental results.

Uniquely combining sketch-based design with Green coordinates processing, our contributions are

- The topology of 3D meshes can be updated dynamically during sketching process, simply by adding or removing key points and curves.
- Smooth free-form surfaces are generated with simple user input and harmonic interpolation.
- A wide range of shape representations can be added to the 2d/3D sketch-based deformation.

The paper is organized as follows. After a review to the related research in Section 2, we provide an intuitive and mathematically elegant solution to modeling the 3D models by simple 3D sketches in 3D space in Section 3. In Section 4 we discuss the 3D feature-preserving deformation with Green coordinates. And the sketch-based image deformation is described in Section 5. Some experimental models created with the new method are detailed in Section 6. Finally, we draw the conclusions and discuss possible future work.

2 Previous work

A large number of the approaches about interactive shape deformation have been proposed in the last decade years. We briefly overview the related works in this section.

Sketch-based interface Current tools for free-form design and subsequent design process, can fall into two categories. One is the professional modeling package which employs parametric patches or subdivision surfaces such as Maya and 3d Max [3, 4]. Users have to firstly lay out the roughest level of patches, then modify control points to add details. It is difficult for a new hand to build control structure from scratch in trying to convey an intended shape, that another group of shape modeling tools [15, 16, 22, 28, 30, 31] have emerged, centering on intuitive modeling metaphors such as sketching, in an effort to hide surface-describing mathematical details from the user. Amendments to the original Teddy system have been proposed recently [21, 22, 28], however, the problem of extending topological diversity of potential model has yet to be addressed. Our system also adopted the Teddy system as the basis for user interface, that said, this system is not constrained topologically, that means it provides multiple interfaces that allows internal structures to be detailed, and aid the user by automatically rotating a model when necessary. The sketch-based interface can also act as an multimedia tools for image composition and clothing design [1, 36].

Image warping Image deformation usually involves the use of handles, with point-shapes [6], lines [5] and polygon grids [26]. By modifying the position and direction of handles, intuitive deformation can be achieved. Igarashi et al. [17] present an interactive system in which a user can move and deform a 2D object without having to build a skeleton or freeform deformation manually. The shape is represented by a triangle mesh, and the user moves several vertices of the mesh as constrained handles. The approach successfully conveys a sense of rigidity of the shape, which is difficult in space-based warping approaches. Schaefer et al. [29] put forward an image deformation method based on linear combination of Moving Least Squares (MLS). The method requires precomputing the deformation functions, and the deformation time is linear with the number of vertices of the sample grid. Weng et al. [35] use sketches to deform images on graphics processing unit (GPU) for real-time performance.

Cage-based geometry deformation Multiresolution approaches can intrinsically preserve local geometric details by frequency decomposition [13, 23]. To preserve surface details, some methods [25, 37, 39] optimize gradient-based or Laplacian-based energies. These surface-based deformation methods restrict target deformed model to manifold meshes in their deformation. Radial basis function (RBF) is another useful tool for translating deformations from control points to embedded models smoothly [7, 33]. Other researchers used closed polygon mesh to embed the deformable model [14, 18, 19]. Floater [10] extended 2D mean value coordinates (MVC) for parameterizations. Ju et al. [20] and Floater et al. [11] extended 2D MVC to 3D space, and used closed 3D cages to deform the target objects. Derose et al. [8] proposed the harmonic coordinates (HC), which was later adopted by Joshi et al. [18] for 2D and 3D character articulation. Green coordinates [24] ensure high quality conformal deformations with a simple cage. Meng et al. [27] generalize the MVC, HC and GC into the concept of Cage Coordinates for image deformation. Laplacian deformation has been also used in facial animations recently [34].

3 3D sketch-based modeling

Prior to performing deformation with using Green coordinates, we first need to generate the 3D surfaces from 2D sketches. It is impractical to automatically extract plausible 3D profiles, without referring to actual topological analysis. This is because 3D models usually have complicated profiles. We generate an artistic 3D curve from a 2D curved sketches, to address this issue by using curvature minimizing depth interpolation. And curve manipulation techniques are used to modify our computed 3D curve. The method under discussion seeks to facilitate the user through 2D interface, reducing overall sketching time for an object.

3.1 Sketching curves

In our system, a given 2D curve is regarded as the projection of a 3D curve with minimum curvature. We therefore propose an efficient algorithm capable of finding a close approximation of this minimum curvature. The resulted 3D curve, through a close approximation of the minimum curvature, produces a rough, high-frequency depth value across the critical points. A *Laplacian filter* is used to generate final space curve, because it delivers smooth curve in a fast and simple way which is an inherent requirement of interactive sketching application. We give a fast and simple method for finding the depth of the sketched 2D curve points that approximates the minimum curvature 3D curve. We assume that the depth values of the control points are equally spaced and the closest point on the curve from any control point has the same depth value as the control point. Hence if we identify these key points on the curve that are closest to the control points and space their depth values equally, we will have the first approximation of the space curve. Even though we do not have an explicit Bezier representation of the 2D sketched curve, the number of critical points in the curve is the lower bound on the number of control points of the hypothetical Bezier representation of the same curve. The critical point need not be the closest curve point (key point) to the corresponding control point. The fundamental source of approximation in our approach comes from the fact that we assume that these critical points are key points.

After 2D-3D transformation, we knits the 3D curves into a network (graph), which is used to directly control the generation of free-from surfaces. In this process, end points of these 3D curves are called linkage points, and the connectivity between linkage points can be defined on a 2D interface. Meanwhile, automatic face recognition is employed to change the resulted topology of control graphs accordingly. Figure 2 gives an illustration of the *Bidirectional Patch Table*(BPT) structure for a tetrahedron. It can be shown that the BPT structure used in computational geometry can be converted from one to the other in the linear time. This implies that the computer space used by a BPT structure to represent a mesh modeling is linear in the size of the mesh, which is the best possible.

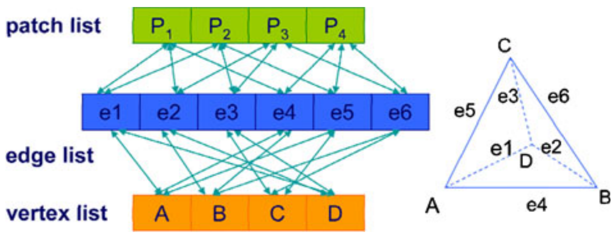


Fig. 2 The Bidirectional Patch Table (BPT) structure for a tetrahedron

Specifically, the proposed system allows the user to outline desired contour of an object, then it will help the user identify faces of the surface in an “intelligent” way. It is noted that the system supports a variety of models including smooth and sharp boundaries, here, the designed meshes can be divided into simpler surfaces, called *patches*. So 3D curves are combined into inter-connected boundaries of these faces. Face recognition is theoretically a face-tracing process in the graph represented by curve networks. For the system to automatically recognize faces on the curve network and quickly triangulate faces, a graph rotation system is applied on the nested curves.

3.2 Surface construction

Once the face boundary connectivity is known, the system generates a smooth freeform surface based on a given boundary curve network by applying harmonic interpolation and the surface refining. There are two steps in generating meshes for 3D profiles:

1. *Topological Triangulation*–Given a closed sequence of connected curves, this step constructs a topology of the triangulation that consists of input vertices on the curve and the newly introduced vertices in the interior of the identified face.
2. *Harmonic Interpolation*–The triangle mesh is automatically extended to the shape that represents the freeform surface by harmonic interpolation. Afterwards, mesh refining is processed to reduce the geometrical noise in generated meshes.

Specifically, in step 1, boundary curves of the identified faces from in previous process are projected onto a plane that best approximates the curve points. Hence the triangulation method is applied on 2D domain, based on the *Delaunay triangulation*, where a triangles is created by joining nearest neighboring nodes at its edges. Since all vertices are projected on the plane, such triangulation is actually a 2D Delaunay triangulation, called *topological triangulation*. Here, the input is a Planar Straight Line Graph (PSLG), including the vertices of the boundary curves projected onto the plane. Quality mesh generation for a PSLG is a standard computation involving two control parameters, a triangle size tolerance (μ) and a minimum angle tolerance (θ). See [12] for descriptions of this computation. The modeler uses freely distributed program Triangle by [32] for this task.

Then mesh nodes generated from triangulating the surface domain with the boundary curve constraints in 2D are translated into a 3D shape by using the transformed profiles in 3D. Mesh vertices are interpolated in a way that conforms to the transformed 3D profiles using Laplacian interpolation. We define the parametric surface by the function $g(u, v) = (x(u, v), y(u, v), z(u, v))$ representing the 3D surface mesh shape, where r and s are the parameters for a position in the 2D plane domain Γ. We now specify $\partial\Gamma$ be the outline of the faces projected onto the 2D plane, and let Γ be the original 3D surface profiles which is interactively modeled and automatically captured before. There are two kinds of boundary conditions applied to the domain Γ, the one is a *Neumann* boundary condition defined at $\partial\Gamma$ because $\partial\Gamma$ is considered as free, and the other is a *Dirichlet* boundary condition defined at Γ, which are illustrated in Fig. 3. Let $g(r, s)$ be the forced boundary condition at Γ,

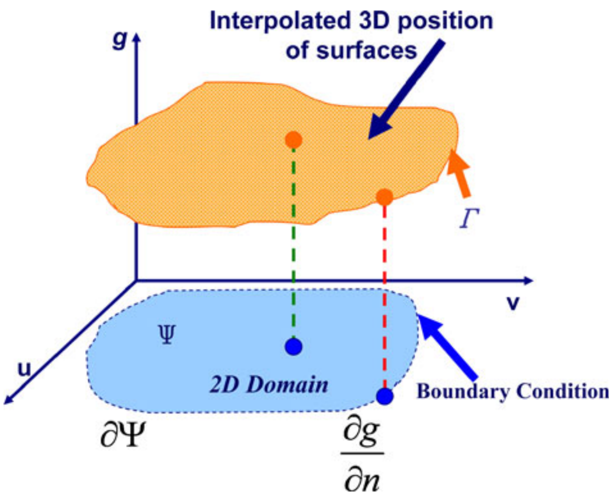

Interpolated 3D position of surfaces
g
v
u
Γ
Ψ
2D Domain
Boundary Condition
$\partial\Psi$
$\frac{\partial g}{\partial n}$

Fig. 3 Interpolation of 3D surface mesh coordinates

which is given by the 3D curves generated interactively as discussed previously. The interpolation problem is mathematically defined as follows.

$$\nabla^2 g = \frac{\partial^2 g}{\partial u^2} + \frac{\partial^2 g}{\partial v^2} = 0 \qquad \text{for } (u, v) \in \Psi \tag{1-1}$$

$$g(u, v) = g(u, v) \qquad \text{for } (u, v) \in \Gamma \tag{1-2}$$

$$\frac{\partial g}{\partial n} = 0 \qquad \text{for } (u, v) \in \partial\Psi \tag{1-3}$$

Functions that satisfy Laplace's equation, $\Delta G = 0$, are called *harmonic functions* [9]. The Laplace's equation is interpreted as partial differential equations governing an elastic membrane problem. The function f specified by (1) can be viewed as 3 harmonic functions defined in Ψ that interpolates $b(u, v)$ on Γ. In implementation, the first step is to discretize domains into simple geometrical shapes like triangles. Parameters u and v represents the coordinates of the vertices in the discretized 2D domain. The coordinates of the transformed profiles in 3D space are treated as the discretized values of b at Γ.

Now the Laplacian interpolation problem of the surface mesh can be solved by a simple hierarchical finite difference solver, though in principle any solution method for Laplace's equation, such as a finite element method, could be used. We adopt the non-hierarchial version of the solver. For each vertex c of the cage, we approximate $g(u_c, v_c)$ over the interior of the cage as follows:

1. Volume grid construction: we set up a regular volume grid that is big enough to enclose the 3D sketched network. We choose the grid to contain 2^s cells on a side. The value of s is the parameter to determine vertex density of the generated mesh surface, which can be interactive adjusted by users.
2. Scan-conversion of triangle surface: scan-convert boundary conditions into the grid, marking each scan converted cell as the surface cell. In our 3D surface

construction, our implementation is currently restricted to triangular faces, meaning that the boundary values varying in a piecewise linear fashion. We therefore use a simple voxel-based triangle scan-converter in this stage.

3. Laplacian smooth: for each surface cell (except the cell is located on the pre-defined curves), replace the position value of the cell with the average of the value of its neighbors. The surface cells are considered to be 6-connected. This Laplacian smoothing step is performed iteratively until the termination criterion is reached.

The solver described above can be significantly accelerated by noting that Laplace's equation produces very smoothly varying functions. By first solving the problem at a lower resolution, better starting points for the iteration can be obtained. The hierarchical solver exploits this observation by "pulling" the boundary conditions up to a coarser level, recursively solving there, "pushing" the coarse solution down to the finer level, then iterating the Laplacian smoothing step until convergence is reached. Now the triangle vertex interpolation is solved by projecting the planar triangles onto the generated surfaces.

4 3D deformation with green coordinates

After obtaining 3D geometrical surface, we use Green coordinates [24] to interactively deform sketch-based models. Green coordinates can be used over convex and concave polygons and polyhedra and, moreover, being based on harmonic functions, produce a conformal mapping in 2D and a quasi-conformal one in 3D. Such a choice allows for non-linear deformation energy only in terms of a few cage vertex positions. We first briefly reviews Green coordinates, then introduce an energy term to penalize isotropic scale variance for better deformations. Because Green Coordinates induces quasi-conformal deformation results in 3D, it will preserve the local shape details intrinsically. The deformation quality of this subspace is much better than the linear spaces (see Fig. 4). Besides of that, it means that we do not have to adopt complex non-linear energy terms for the shape preserving property and the energy term for solving the cage vertices' displacements could be very simple and solved fast, as shown in Fig. 5.

Following the denotations in [24], let the cage be an oriented simplicial surface, that is $P = (\mathbb{V}, \mathbb{F})$, where $\mathbb{V} = \{v_i\} \subset \mathbb{R}^d$ are the vertices and $\mathbb{T} = \{f_j\}$ are the simplicial face elements. $n(f_j)$ is denoted as the unit outward normal to the oriented

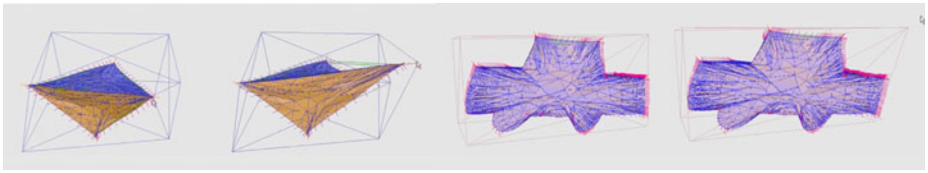


Fig. 4 The cage-based deformation of 3D models by using Green coordinates: (*left*) deformation of a quadrangle model; (*right*) deformation of a car model

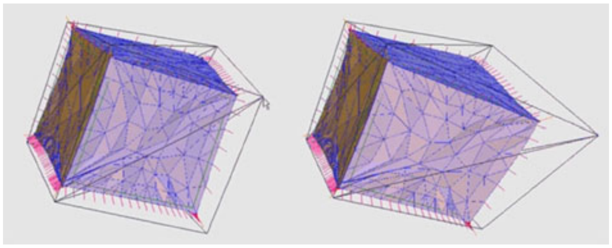


Fig. 5 The sketch-based geometry deformation with Green coordinates

simplicial face $f_j(\|n(f_j) = 1\|)$. The deformation function on each interior point κ about the deformed cage C is described as below:

$$\kappa \mapsto D(\kappa; C) = \sum_{v_i \in \mathbb{V}} \phi_i(\kappa) v_i + \sum_{f_j \in \mathbb{F}} \upsilon_j(\kappa) s_j n(f_j), \tag{2}$$

where s_j is the scaling factors. We change the summarizing into matrix form and pack the variables:

$$D(\kappa) = \Phi(\kappa) V + \Upsilon(\kappa) N, \tag{3}$$

where V and N are the column vectors of packed vertices coordinates v_i and scaled normal vector $s_j n(f_j)$ respectively. It must be noticed that N is non-linearly depended on V.

4.1 Deformation constraint

Due to the fact that the degree of freedoms (DOFs) of the sketched cage are very high, Green coordinates deformation may lead to degenerated optimization problem. Specifically, the quasi-conformal deformation field may contain highly variant scaling in different region (Fig. 6a).

To prevent the deformation scale changes too vapidly, the following linear least square is used to calculate the variation of deformation gradient at point κ in the cage:

$$\|\nabla^2 D(\kappa)\|_D^2 = \left\| \frac{\partial^2 D(\kappa)}{\partial^2 \kappa} \right\|_D^2, \tag{4}$$

where $\| \cdot \|_D$ denotes the Frobenius norm.

Note that the secondary derivation of the deformation function $\nabla^2 D(\kappa)$ does not converge at the point η where is closing to the vertices of the cage. So it is difficult to integrate energy on all over-the-cage regions. Let $\mathbb{C}$ denotes the region of the cage polygons and we shrink the region by a distance l to the region $\mathbb{C}'$. The total energy is defined as the integration of the shrinked region:

$$\begin{aligned} E_{\text{distortion}} &= \int_{\mathbb{C}'} \left\|\nabla^2 D(\kappa)\right\|_D^2 l\kappa \\ &= \int_{\mathbb{C}'} \left\|\left(\nabla^2 \Phi(\kappa)\right) V + \left(\nabla^2 \Upsilon(\kappa)\right) N\right\|_D^2 l\kappa \\ &\triangleq \int_{\mathbb{C}'} \left\|\hat{\Phi}(\kappa) V + \hat{\Upsilon}(\kappa) N\right\|_D^2 l\kappa. \end{aligned} \tag{5}$$

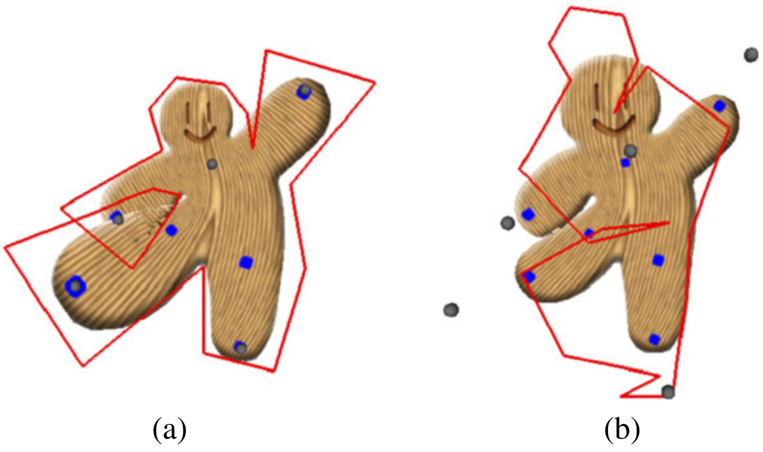


Fig. 6 **a** Without distortion constraints, the deformation (constrained by deformation gradients) is often odd because the scale may rapidly changed in the cage. $E_{\text{distortion}}$ penalizes such variation of scale and leads to the result in **b** with nearly uniform scaling

Rigid translation, rotation and uniform scaling are not enough to increase $E_{\text{distortion}}$. Linear least squares (5) can efficiently eliminate odd results as shown in Fig. 6b from the conformal deformation space defined by the Green coordinates.

4.2 GPU implementation of green deformation

The Green deformation works in two steps: (1) preprocessing: the Green coordinates with respect to the initial sketched cage vertices and face normals are computed and recorded for the deformable mesh vertex of the object; (2) deformation: every time that the sketched cage is modified, the mesh inside it is recalculated using the new cage's geometric information. We calculate the Green coordinates of 2D /3D objects and record them in graphics hardware (GPU) memory. Specifically we use two-pass GPU shader program to achieve the realtime image deformation.

1. Preprocessing of Green Deformation: the computed Green Coordinates of the vertices of the mesh are stored in a texture array of floats of 32 bits in the CPU. To be more precise, if the cage has N vertices and M faces, we have $N + M$ Green Coordinates for each vertex of the mesh. To store them, we need k RGBA textures, where $k = \lceil (N + M)/4 \rceil$. Once we know how many textures will be used, we have to store the Green Coordinates of the i-th vertex of the mesh in the ith pixel of these textures. For example, imagine we have a cage with $N = 28$ vertices and $M = 38$ faces; this means that we need a texture array of $k = 17$ RGBA textures.
2. Green Deformation step: every time the cage is deformed we compute the deformation in the GPU in a two-pass algorithm. First, a shader reads the textures created in the preprocess step and, using (2), calculates the new mesh. The new positions of the vertices are stored in another texture (see Fig. 7) and the information is passed to the second step through a Frame Buffer Object.

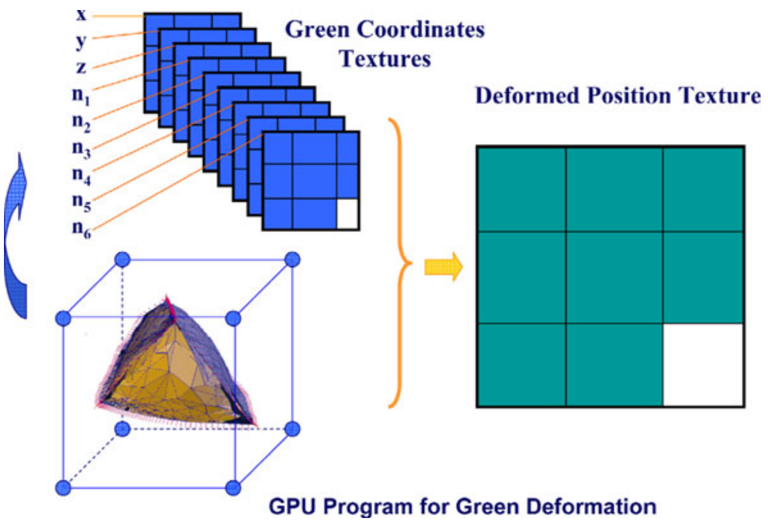

x
y
z
n1
n2
n3
n4
n5
n6
Green Coordinates Textures
Deformed Position Texture
GPU Program for Green Deformation

Fig. 7 GPU implementation of Green deformation

> More precisely, in order to get the new position of the *i*-th vertex, we multiply the first control vertex with the *R* component of the first texture, the second control vertex with the *G* component of the same texture, and so on. Then, we compute the sum of all this products and the result is stored in the *i*-th pixel of a new *RGB* texture of floats of 32 bits. This new texture is used to initialize the Frame Buffer Object which will be used as a Vertex Buffer Object to render the geometry in the second render pass.

With this GPU-based implementation, we can deform medium-sized models in real time. It also presents the comparison between doing the deformation in the GPU or doing it in the CPU. We can observe that if we use the GPU implementation we can accelerate the deformation process between 50–80 times.

5 Sketch-based image deformation

Green coordinates can be also used to deform a image region enclosed by a user-sketched bounding box. Traditional approaches deform an image-based object by selecting some handles from 2D plane to control warping effects. Recently researchers focuses more on deforming objects by cages, a low polygon-count polyhedron, typically with a shape similar to the object. Specifically, a 2D cage is an enclosed polygon in an image, surrounding the region-of-interest (ROI). In the 2D sketch-based deformation, we can first enclose the region-of-interest with 2D sketched curves, then simplify the closed sketch curves to from a cage. After the curve simplification, the cage, enclosing a 2D region of an object, is in effect a low polygon-count bounding box. For a point *p* inside the cage, if its position can be

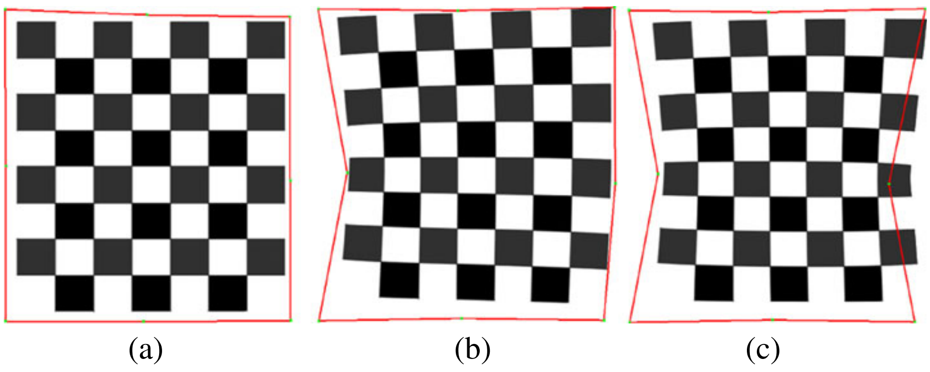


Fig. 8 Image deformation using Green coordinates

denoted by linear combination of the cage vertices' positions and cage faces' normals, there is a general equation [27]:

$$p = \Sigma_{i \in C_v} \phi_i(p) v_i + \Sigma_{j \in C_f} \upsilon_j(p) n(f_j) \quad (6)$$

where v_i is the $i-th$ cage vertex position, and $n(f_j)$ is the normal of the $j-th$ face. The coefficients $\phi_i(p)$, $\nu_j(p)$ are given by using Green coordinates in 2D space (see [24]):

$$\phi_i(\kappa) = \int_{\xi \in N_{\{v_i\}}} \Gamma_i(\xi) \frac{\partial G(\xi, \kappa)}{\partial n(\xi)} d\sigma_\xi$$

$$\nu_j(\kappa) = -\int_{\xi \in t_j} G(\xi, \kappa) d\sigma_\xi$$

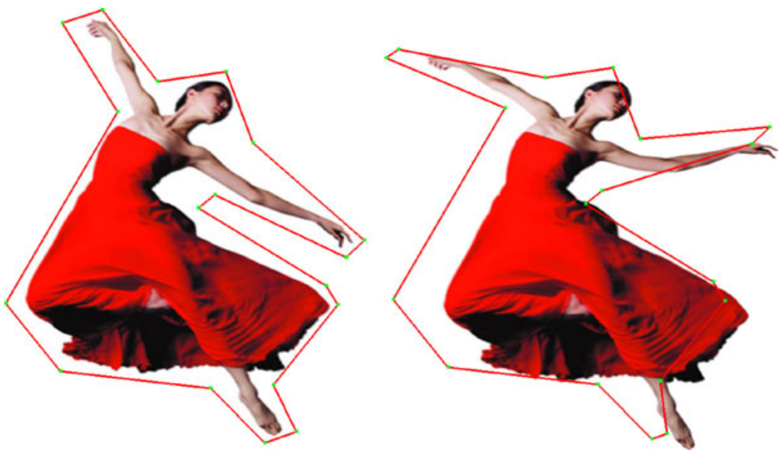


Fig. 9 2D cages for image deformation based on Green coordinates: (*left*) original image and its cage; (*right*) deformed image and its cage

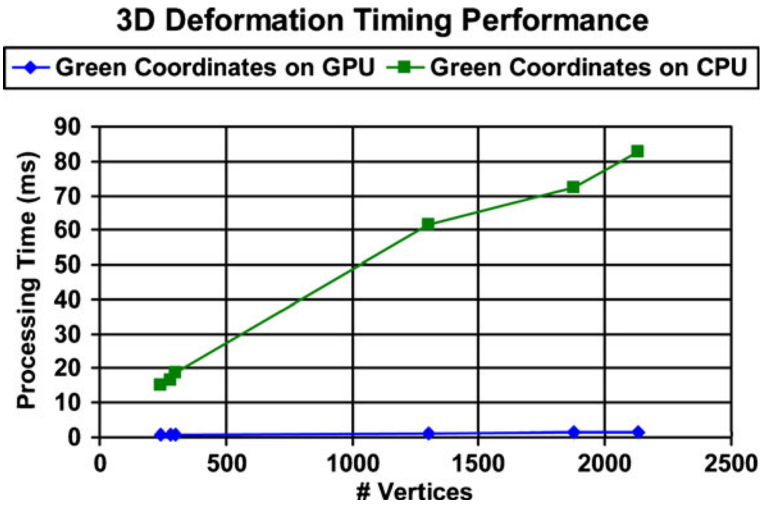


Fig. 10 Comparison between processing times of the Green deformation in GPU and in CPU

where $G(\xi,\kappa) = \frac{1}{2\pi} log\|\xi - \kappa\|$, and Γ_i is the piecewise-linear hat function defined on $N\{v_i\}$ which is one at v_i and zero at all the other vertices. We use a sketch-based cage to deform the image region, the chessboard region of image will be deformed as shown in Fig. 8. The user can interactively move cage vertices, so that the content inside ROI will be modified by the computation of Green coordinates (see Fig. 9). The whole interactive process on the image plane is entirely order-independent and

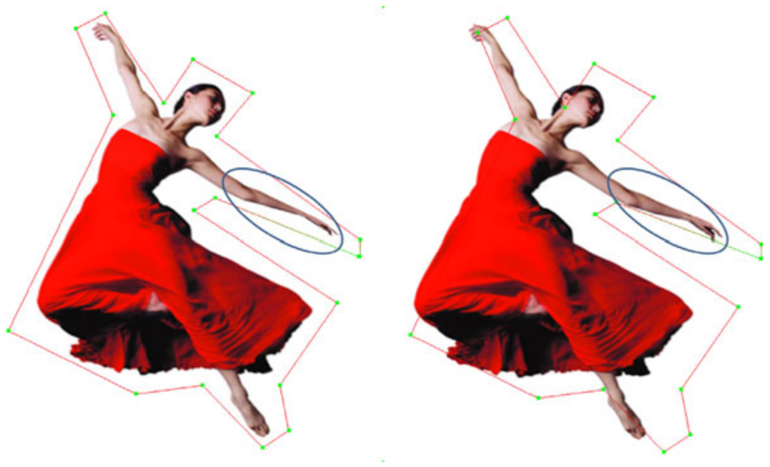


Fig. 11 Image deformation comparison: (*left*) deformation with mean value coordinates; (*right*) image deformation with Green coordinates

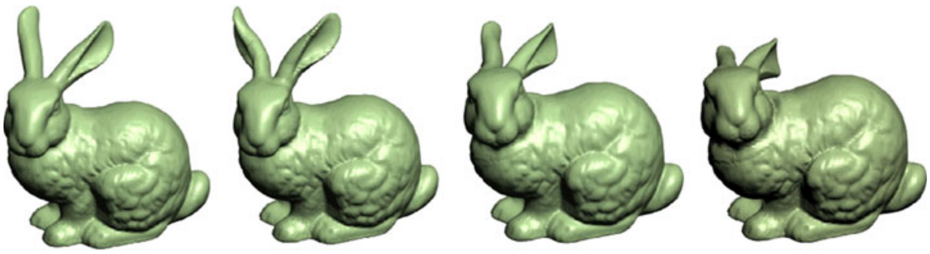


Fig. 12 Cage-based Green deformation of the bunny model with GPU acceleration

can be implemented on GPU in a parallel manner, for real-time deformation (Figs. 10 and 11).

6 Experimental results

The proposed approach is developed on a 2.6 GHz Intel Core CPU, together with nVidia GeForce 8800 GPU and 3G RAM. The GPU program described above are implemented by NVIDIA's Cg program.We have tested our approach generating a number of simple models. Figures 12, 13 and 14 show some of the geometry and image deformation produced by using our interactive method. Table 1 summarizes the experimental performances of our 3D modeling and deformation. Our system provides a simple interface where the user can draw lines, rotate the view and continue drawing until the desired shape is achieved. With the GPU implementation of Green coordinates, we can deform the middle-scale geometric models in realtime. Figure 10 shows the timing performance of our approach. We find that the GPU/CPU speedup ratio and parallel efficiency are improved with the growth of the geometric complexity. This has implications for the use of GPU implementation in large models.

Meanwhile, various image deformation examples are shown in different resolutions, for our test, even the images with millions of pixels can be well deformed in realtime. Please note that our sketch-based image deformation doesn't need to sample the image into a lower resolution, as the approach used in [29]. Table 2 shows the timing performance results of interactive image deformation by our approach. Due to the GPU parallel implementation of Green coordinates computation, our defamation process can be updated and displayed in realtime. Figure 13 shows our image deformation over the ROI to generating plausible results. Since the image deformation using Green coordinates is processed in realtime, we can simply set the cage vertices' moving velocities to generate 2D animation. Figure 14 shows the consequent 2D dancing animated image sequence by using Green coordinates. Comparing to the deformation using the mean value coordinates, our image deformation preserves the local detail of the ROI well, due to the GPU-based implementation Green coordinates. Figure 11 shows that the deformation with mean value coordinates cause the serious distortion in the dancer's arm, while our approach preserve such region well.

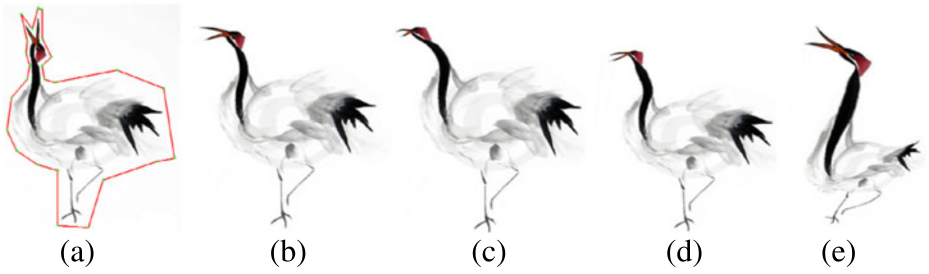


Fig. 13 Deformation of the Chinese painting picture of the red-crowned crane

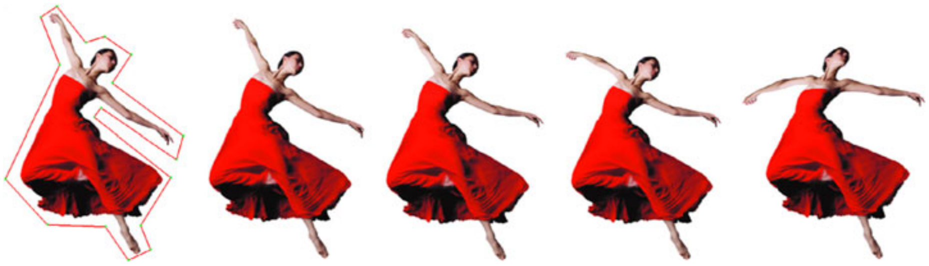


Fig. 14 Generate the animation automatically after the 2D cage vertices' velocities are interactive set by users

Table 1 This table lists the modeling times, data sizes, and deformation times (FPS) of the 3D sketched-based models

Model	#Curves (input)	#Vertices	#Triangles	Modeling time (s)	GPU deformation time (ms)
Cube	6	239	416	5	0.56
Pyramid	8	298	518	6	0.78
Car	36	282	524	6	0.65
House	15	1,304	4,604	25	1.26
Tree	20	1,874	3,740	24	1.31
Leaf	10	2,132	4,436	32	1.38

Modeling times (seconds) measure the total time beginning with reading the input sketches and ending with writing the triangle mesh

Table 2 Processing time statistics

Image	Image size	#Cage vertex	Update without CPU	With GPU	
				Preprocessing time	Updating time
Dancing	283 × 400	18	0.092 s	1.7278 s	≤0.001 s
Butterfly	490 × 345	17	0.318 s	2.3385 s	≤0.001 s
Chessboard	443 × 501	8	0.417 s	1.6089 s	≤0.001 s
Crane	288 × 432	25	0.237 s	2.4676 s	≤0.001 s
Giraffe	332 × 500	22	0.353 s	2.8920 s	≤0.001 s

7 Conclusion and future work

In this paper, we propose a new sketch-based deformation using Green coordinates. The 3D space curve generation uses a single view information about a 2D curve to find the curvature minimizing 3D curve. In addition, we combine the graph rotation system and the harmonic interpolation for sketch-based modeling. We have also introduced the Green Coordinates for our sketch-based deformations, providing shape-preserving mappings from geometric models and image contents. During the 2D Green deformation, we can deform the Region-of-Interest marked by a sketched cage in real time, and modified the cage to make the detail-preserving deformation with the graphics hardware implementation GPU.

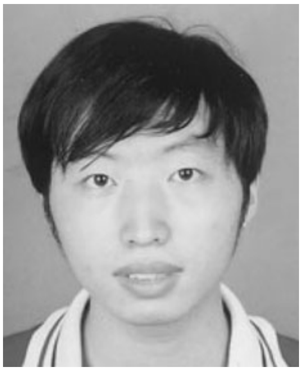
Planned future work includes sketch-based the on-surface vector field manipulations. We are also considering to use Green coordinates for video editing and fluid simulation on GPU. Finally, we believe that developing efficient ways to couple user-guidance with optimization techniques is a fertile ground for future research.

Acknowledgements The authors would like to thank all reviewers for their helpful suggestions and constructive comments, which have significantly improved our paper. Many thanks to Shunbing Chen for implementing and testing many parts of our system, to Dr. Jin Huang (Zhejiang University) for useful discussions. The work is supported by National Basic Research Program of China (973 Program) No. 2009CB320802, 2011CB302200, National Natural Science Foundation of China No. 60833007,60873136, Beijing Natural Science Foundation (Content-Aware Image Synthesis and Its Applications, No. 4112061), French System@tic Paris-Region(CSDLProject) and ANR-NSFC No.60911130368, Hong Kong Research Grants Council grants 416007 and 415806.

References

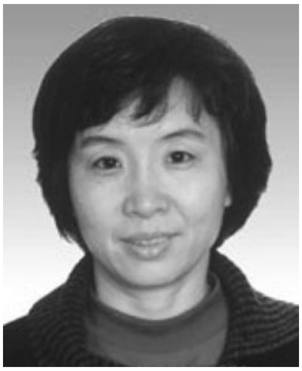
1. Andreou I, Sgouros N (2007) Utilizing shape retrieval in sketch synthesis. Multimed Tools Appl 32:275–291
2. Arts E (2007) SPORE. http://www.spore.com
3. Autodesk (2008) 3DS MAX. http://www.autodesk.com/
4. Autodesk (2008) MAYA. http://www.autodesk.com/
5. Beier T, Neely S (1992) Feature-based image metamorphosis. In: SIGGRAPH '92: proceedings of the 19th annual conference on Computer graphics and interactive techniques. ACM, New York, NY, USA, pp 35–42
6. Bookstein FL (1989) Principal warps: thin-plate splines and the decomposition of deformations. IEEE Trans Pattern Anal Mach Intell 11(6):567–585
7. Botsch M, Kobbelt L (2005) Real-time shape editing using radial basis functions. In: Computer graphics forum, pp 611–621
8. Derose T, Meyer M (2006) Harmonic coordinates. Tech. rep., Pixar Animation Studios
9. Duchamp T, Certain A, Derose A, Stuetzle W (1997) Hierarchical computation of pl harmonic embeddings. Tech. rep.
10. Floater MS (2003) Mean value coordinates. Comput Aided Geom Des 20(1):19–27
11. Floater MS, Kós G, Reimers M (2005) Mean value coordinates in 3D. Comput Aided Geom Des 22(7):623–631
12. George PL, Borouchaki H (1998) Delaunay triangulation and meshing. Hermes, Paris, France
13. Guskov I, Sweldens W, Schröder P (1999) Multiresolution signal processing for meshes. In: SIGGRAPH '99: proceedings of the 26th annual conference on Computer graphics and interactive techniques. ACM, New York, NY, USA, pp 325–334
14. Huang J, Shi X, Liu X, Zhou K, Wei LY, Teng SH, Bao H, Guo B, Shum HY (2006) Subspace gradient domain mesh deformation. ACM Trans Graph 25(3):1126–1134

15. Igarashi T, Hughes J (2003) Smooth meshes for sketch-based freeform modeling. In: Proceedings of the 2003 symposium on interactive 3D graphics, pp 139–142
16. Igarashi T, Matsuoka S, Tanaka H (1999) Teddy: a sketching interface for 3d freeform design. In: Proceedings of ACM SIGGRAPH 1999. ACM, ACM Press/ACM SIGGRAPH, pp 409–416
17. Igarashi T, Moscovich T, Hughes JF (2005) As-rigid-as-possible shape manipulation. In: SIGGRAPH '05: ACM SIGGRAPH 2005 papers. ACM, New York, NY, USA, pp 1134–1141
18. Joshi P, Meyer M, DeRose T, Green B, Sanocki T (2007) Harmonic coordinates for character articulation. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers. ACM, New York, NY, USA, p 71
19. Ju T, Schaefer S, Warren J (2005) Mean value coordinates for closed triangular meshes. In: SIGGRAPH '05: ACM SIGGRAPH 2005 papers. ACM, New York, NY, USA, pp 561–566
20. Ju T, Schaefer S, Warren J (2005) Mean value coordinates for closed triangular meshes. ACM Trans Graph 24(3):561–566
21. Karpenko OA, Hughes JF (2006) Smoothsketch: 3D free-form shapes from complex sketches. ACM Trans Graph 25(3):589–598
22. Karpenko OA, Hughes JF, Raskar R (2002) Free-form sketching with variational implicit surfaces. Comput Graph Forum 21(3):585–594
23. Kobbelt L, Campagna S, Vorsatz J, Seidel HP (1998) Interactive multi-resolution modeling on arbitrary meshes. In: SIGGRAPH '98: proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, New York, NY, USA, pp 105–114
24. Lipman Y, Levin D, Cohen-Or D (2008) Green coordinates. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers. ACM, New York, NY, USA, pp 1–10
25. Lipman Y, Sorkine O, Cohen-Or D, Levin D, Rössl C, Seidel HP (2004) Differential coordinates for interactive mesh editing. In: Proceedings of shape modeling international. IEEE Computer Society Press, Los Alamitos, CA, pp 181–190
26. MacCracken R, Joy KI (1996) Free-form deformations with lattices of arbitrary topology. In: SIGGRAPH '96: proceedings of the 23rd annual conference on computer graphics and interactive techniques. ACM, New York, NY, USA, pp 181–188
27. Meng W, Sheng B, Wang S, Sun H, Wu E (2009) Interactive image deformation using cage coordinates on gpu. In: VRCAI, pp 119–126
28. Nealen A, Igarashi T, Sorkine O, Alexa M (2007) FiberMesh: designing freeform surfaces with 3D curves. ACM SIGGRAPH
29. Schaefer S, McPhail T, Warren J (2006) Image deformation using moving least squares. In: SIGGRAPH '06: ACM SIGGRAPH 2006 papers. ACM, New York, NY, USA, pp 533–540
30. Sheng B, Li P, Sun H (2009) Image-based material restyling with fast non-local means filtering. In: ICIG, pp 841–846
31. Sheng B, Wu E, Sun H (2008) Sketching freeform meshes using graph rotation functions. Vis Comput 24(7–9):745–752
32. Shewchuk JR (1996) Triangle: engineering a 2D quality mesh generator and delaunay triangulator. In: First workshop on applied computational geometry. ACM Press, pp 124–133
33. Sumner RW, Schmid J, Pauly M (2007) Embedded deformation for shape manipulation. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers. ACM, New York, NY, USA, p 80
34. Wan X, Jin X (2011) Data-driven facial expression synthesis via laplacian deformation. Multimed Tools Appl 1–15. doi:10.1007/s11042-010-0688-7
35. Weng Y, Shi X, Bao H, Zhang J (2008) Sketching MLS image deformations on the GPU. Comput Graph Forum 27(7):1789–1796
36. Yu H, Qin S, Sun G, Wright D (2011) On generating realistic avatars: dress in your own style. Multimed Tools Appl 1–18. doi:10.1007/s11042-011-0781-6
37. Yu Y, Zhou K, Xu D, Shi X, Bao H, Guo B, Shum HY (2004) Mesh editing with poisson-based gradient field manipulation. In: SIGGRAPH '04: ACM SIGGRAPH 2004 papers. ACM, New York, NY, USA, pp 644–651
38. Zeleznik RC, Herndon KP, Hughes JF (1996) Sketch: an interface for sketching 3D scenes. In: Proceedings of ACM SIGGRAPH 1996. ACM Press, pp 163–170
39. Zhou K, Huang J, Snyder J, Liu X, Bao H, Guo B, Shum HY (2005) Large mesh deformation using the volumetric graph laplacian. In: SIGGRAPH '05: ACM SIGGRAPH 2005 papers. ACM, New York, NY, USA, pp 496–503



Bin Sheng received his BA degree in English and BE degree in computer science from Huazhong University of Science and Technology in 2004, and MS degree in software engineering from University of Macau in 2007, and PhD Degree in computer science from The Chinese University of Hong Kong in 2011. He is currently an assistant professor in Department of Computer Science and Engineering at Shanghai Jiao Tong University. His research interests include virtual reality, computer graphics and image based techniques.

Weiliang Meng received the B.E. degree in Computer Science from Civil Aviation University of China in 2003, M.Sc. degree in Computer Application from Tianjing University in 2006, and PhD degree in Computer Application from State Key Laboratory of Computer Science at Institute of Software, Chinese Academy of Sciences. He is correctly a postdoctor in National Laboratory of Pattern Recognition (NLPR) at Institute of Automation, Chinese Academy of Sciences. His research interests include geometry modeling, and image based modeling and rendering.



Hanqiu Sun received her MS degree in electrical engineering from University of British Columbia, and PhD degree in computer science from University of Alberta, Canada. She is an associate professor in Department of Computer Science and Engineering, Chinese University of Hong Kong (CUHK). Her current research interests include virtual and augmented reality, interactive graphics/ animation, hypermedia, mobile image/video processing and navigation, tele-medicine, realistic haptics simulations.

Enhua Wu received the BS degree from Tsinghua University in 1970, and the PhD degree from the University of Manchester (UK) in 1984. He is currently a research professor at the Institute of Software, Chinese Academy of Sciences. He has also been teaching at the University of Macau since 1997, where he is currently the associate dean of Faculty of Science and Technology. His research interests include realistic image synthesis, virtual reality, and scientific visualization.