

Relief Extraction and Editing

Yin Chen^a, Zhi-Quan Cheng^{a,b,*}, Jun Li^a, Ralph R. Martin^b, Yan-Zhen Wang^a

^a*School of Computer Science, National University of Defense Technology, China*

^b*School of Computer Science and Informatics, Cardiff University, Wales, UK*

Abstract

Bas-reliefs are widely used in the world around us, for example, on coinage, for branding products, and for sculptural decoration. Reverse engineering of reliefs—extracting existing reliefs from input surfaces—makes it possible to apply them to new items; relief editing tools allow modification of reverse-engineered reliefs. This paper presents a novel approach to relief extraction based on differential coordinates, which offers advantages of speed and precise extraction. It also gives the first method in the literature specifically designed for relief editing. The base surface is estimated using normal smoothing and Poisson reconstruction, allowing a relief (which may lie on a smooth or textured input surface) to be automatically extracted by height thresholding. We also provide a range of relief editing tools, also using differential coordinates, permitting both global transformations (translation, rotation, and scaling) of the whole relief, as well as local modifications to the relief. Our

*Corresponding author. Tel.: +86 (0)731-8457-5993;

Email addresses: Chris.leo.chan@gmail.com (Yin Chen),
Cheng.zhiquan@gmail.com (Zhi-Quan Cheng), Jun.johnson.li@gmail.com (Jun Li),
Ralph@cs.cardiff.ac.uk (Ralph R. Martin), Yanzhen.wang@gmail.com (Yan-Zhen Wang),
Yukun.Lai@cs.cardiff.ac.uk (Yu-Kun Lai), Gangdang@nudt.edu.cn (Gang Dang),
Syjin1937@163.com (Shi-Yao Jin)

relief editing algorithm, unlike generic mesh editing algorithms, is specifically designed to preserve the geometric detail of the relief over the base surface. The effectiveness of our methods is demonstrated on various examples of real industrial interest.

Keywords: Bas-relief, Relief extraction, Relief editing, Differential coordinates, Laplacian smoothing, Poisson reconstruction

1. Introduction

Bas-reliefs take the form of low-height details, similar to a raised (or, less commonly, lowered) picture, applied to an underlying base surface. They are widely used in daily life to decorate or identify man-made objects, e.g. to apply branding to merchandise, as sculptures on coins and porcelain, and as architectural decoration. The production of reliefs is currently a costly and time-consuming process, requiring skilled sculptors and engravers. Reverse engineering tools which perform automatic extraction of a relief from a scanned model, for application to a new surface, or which can edit it in place to modify its shape or location, could greatly reduce production time. They also have the potential to produce more accurate results than ad-hoc craft methods currently in use.

Providing a simple and robust method for relief extraction is challenging. While at times the relief may lie on a simple base surface such as a plane, a cylinder, or a surface of revolution, which can readily be estimated, in many other cases the base surface is unknown [1]. Starting from an initial user-drawn contour loosely enclosing the relief (a *snake*), Liu et al. [1, 2, 3, 4] gave a series of algorithms to separate reliefs from their underlying

surfaces, both for smooth and textured backgrounds. Zatzarinni showed how to automatically extract reliefs by defining a height function along the base surface normals, the latter being adaptively estimated using local geometric features computed via experimentally determined coefficients [5]. Our paper provides an alternative method to automatically extract reliefs via recovery of the underlying base surface.

Shape editing is a fundamental topic in geometric modeling and processing, with many approaches both for global operations [6] and local modification [7]. However, unlike the arbitrary manipulations available to a free-standing model, relief editing is strongly constrained by the notion that the relief is a modification of an underlying base surface. Existing editing methods are thus not directly applicable, and new tools are needed specific to relief editing. We provide such a set of user-guided tools, with a simple and intuitive interface. These include tools for global transformation, allowing translation, rotation, and isotropic scaling of the relief over the base surface, as well as a local deformation tool for modifying relief shape detail. Our approach also lends itself to transfer of reliefs to new surfaces. Our tools operate directly on mesh models of the kind acquired by laser-range scanners, allowing them to be used in a reverse engineering workflow.

The main contributions of the paper are (i) a novel algorithm for relief extraction, which is faster than previous methods, yet times more precise in its results, and (ii) the first tools in the literature specifically designed for editing reliefs. Example uses of our methods are demonstrated in Figure 1. Our relief extraction approach estimates a base surface using normal smoothing and Poisson surface reconstruction, followed by height thresholding. The

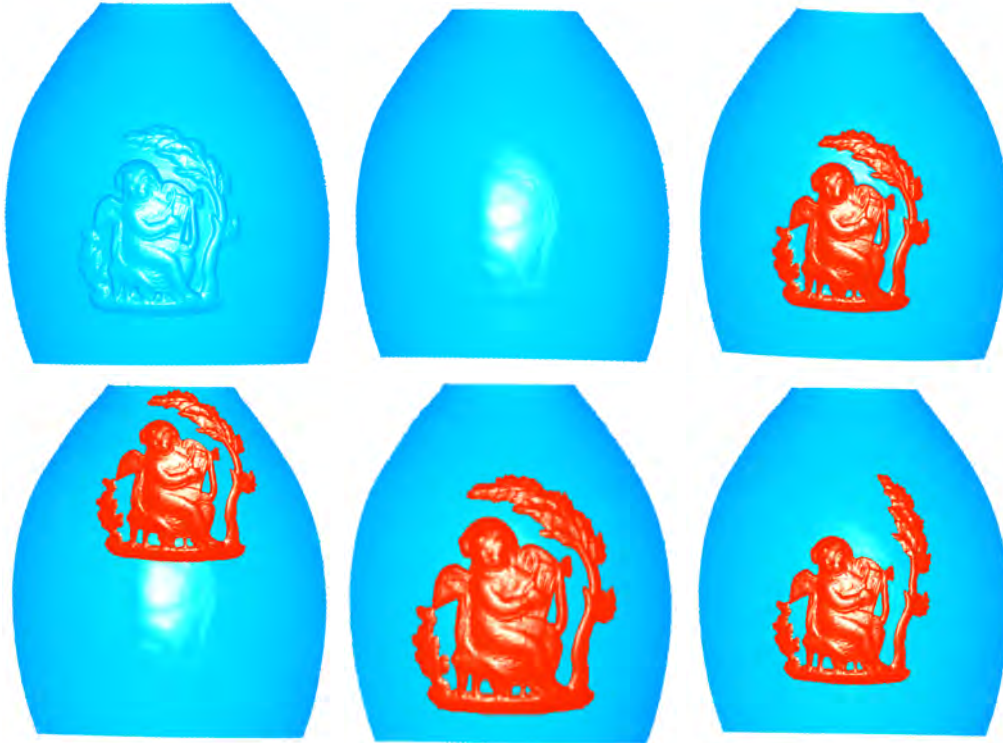


Figure 1: Relief extraction and editing are performed in the gradient domain. An acquired model (top left) is processed to extract the relief (top right) via estimation of an underlying base surface (top center) using Poisson-based smoothing. The extracted relief may be edited in place on the input model, e.g. moving it as a whole across the background surface (bottom left), or locally changing the detail (bottom right).

editing tools are based on the use of *differential coordinates* [8, 9] (Laplacian coordinates), which encode geometric details. While mesh editing methods based on differential coordinates already exist [8, 9], our new tools impose constraints specific to bas-relief editing. In particular, to ensure high-quality results, we preserve the height of relief features over an unchanged base surface, and avoid distortion while modifying geometric details of the relief.

To perform relief extraction, we estimate the underlying curved base sur-

face using a smoothing process. We smooth surface normals to provide guidance vectors for Poisson surface reconstruction to collapse the relief to the level of the base surface. This Poisson problem can be represented as a sparse linear least-squares system which can be efficiently solved. This base surface estimation process is a global smoothing process, and so is both robust and insensitive to noise, and furthermore allows us to extract reliefs superimposed upon textured backgrounds. Having estimated the base surface, we can measure the height of each surface point and use thresholding to extract the relief. Relief extraction is further explained in Section 3.

Differential coordinates are also employed in our relief editing system which directly manipulates the extracted relief over the estimated base surface and original mesh. Global and local editing are easy to control by locally manipulating handles attached to the relief. We place special emphasis on providing an intuitive user interface, robustness and speed, while at the same time being careful to avoid distortion in the final results. Details of these editing tools are provided in Section 4.

2. Related work

2.1. Relief extraction

The first work on relief detection and extraction can be traced back to [10], which decoupled a cuneiform tablet into a smooth B-spline base and a displacement map capturing the inscribed marks. Liu et al. [2] similarly used a B-spline fitting algorithm to estimate the base surface underlying a relief. The position of the relief is extracted starting from a snake loosely drawn by the user around the relief, which snaps to the relief boundary [1]. Snakes can

also be used to separate geometric reliefs from *textured* backgrounds, after texture classification or alternatively surface smoothing [3]. Further work has identified *periodicity* in reliefs extracted by these earlier methods, finding a single repeat unit by determining correspondences between adjacent repeats using an iterative closest point algorithm. A different approach to relief extraction by Zatzarinni [5] requires no user input other than control parameters; global optimization estimates the height of each vertex via estimation of base surface normals. We give a further automatic relief extraction algorithm, in which we reconstruct the base surface by considering smoothed normals, and separate the relief from the surrounding background.

2.2. Differential coordinates and editing

There are many papers on mesh editing based on the use of differential coordinates. In the following we review only representative methods, and refer the reader to the comprehensive surveys in [8, 9].

Discrete Laplacian coordinates, first proposed by [11], have been further improved by adding local rotation estimation [12], transformation linearization [13], the provision of a sketch-based interface [14], generalization to a volume graph Laplacian [15], and affine transformation using a two-step strategy [16]. Laplacian coordinates permit gradient-based editing concurrently with other Laplacian-based techniques. The first such mesh editing algorithm modified the original mesh geometry implicitly through gradient field manipulation [17], using a geodesic interpolation method to propagate user specified rotation and scaling constraints determined at given handles (vertices) to the whole mesh. Other approaches for transformation constraint propagation are based on interpolation [15], harmonic functions [18], or ma-

terial properties [19]. Gradient deformation has also been applied to deformation transfer [20] and mesh morphing [21].

In addition to the above linear variational algorithms, nonlinear optimization methods have also been used to determine transformations of Laplacian coordinates and vertex coordinates from pure translations of deformation control handles. The Gauss-Newton method has been used for both a subspace gradient deformation algorithm [22] and a dual Laplacian coordinate algorithm [23]. Later an alternative linear least-squares deformation solver was proposed [24] based on a particular rigid motion representation [25], in order to provide better deformations in cases which subspace methods find difficult. Handle-aware isolines have been employed to reduce the nonlinear optimization problem to one over affine transformations, rather than directly involving vertex coordinates [26].

Although relief editing has not been explicitly considered in previous papers, techniques relevant to relief editing have appeared. Geometry images have been used for texture or geometric detail transfer [27], as have encoded differences of Laplacian coordinates between an original surface and a smoothed, low-frequency version, together with an inverse Laplacian transform [13]. Correspondences between the source and target surface regions must be manually established to perform the transfer, and in general, this is difficult for the user to achieve, especially if the geometries differ significantly. Such manually-specified correspondences can also be used in a Poisson editing framework to merge meshes to construct a new object [17]. However, it is infeasible for a user to interactively establish the large number of accurate correspondences needed for relief transfer. Thus, Zatzarinni [5] showed how

to paste an extracted relief onto another target area by merging the parameterized domains of the two surfaces. This requires much less user interaction, as correspondences can be determined by choice of boundary conditions for the parametrization algorithm. However, it still provides an awkward interface for relief editing, as the user has to draw a target shape similar to the source region. This is difficult in practice, even if the user just wishes to arbitrarily translate the relief to another position on the same model.

An alternate is cut-and-paste parametrization-based editing, which provide a simpler method for the user to indicate the destination region [28]; however, the user still has to draw a branched curve to serve as the spine of the relief.

Instead, inspired by a position-based dynamic simulation algorithm [29], we provide a relief editing system with a direct and simple interface. The user just drags a few manipulation handles to achieve intended results, such as global translation, rotation, or scaling of the whole relief on the background, or directly modifying local details of the relief. Our approach is conceptually simple and easy to implement, providing interactive performance even without GPU acceleration.

3. Relief Extraction

We start from a captured mesh which includes the relief, surrounded by background surface. Let this triangle mesh be $M = \{V, E, T\}$, where V is the set of N_V vertices, E is the set of edges, and T is the set of triangles. We must first automatically separate the part R of the mesh which belongs to the relief from the background surface G (which may be smooth or textured):

i.e. M should be partitioned into R and G . We assume that the relief can be described as an offset relative to an underlying smooth base surface B . As B is smooth, then its normals also vary smoothly. Thus, to perform relief extraction, we first estimate these smooth normals and then reconstruct B using a Poisson equation in which the boundary conditions come from the background surface surrounding the relief. By finding the height of each mesh vertex above this base surface, we then determine R as the union of all vertices that are higher than a threshold. Figure 2 illustrates how this works both for smooth and textured backgrounds. We first perform Laplacian smoothing of the *vertex positions* of the whole mesh M , to give a new mesh M' . We now compare the normal of each vertex in M with the corresponding vertex normal in M' . Ones which are similar are considered to be reliable (i.e. to represent the correct normal direction). Next, we smooth the *normals* of M using Laplacian smoothing, but build in a constraint to preserve the normals at reliable vertices. This gives a smooth normal field over M which represents the normals of the overall shape. From this normal field, we carry out Poisson reconstruction following [17], using the boundary of the region of interest as position constraints—the relief is presumed not to extend to the region boundary. This gives the base surface. Finally, the relief is detected as those regions of M which have heights greater than some threshold above the base surface.

We briefly note that M' is itself unsuitable as a base surface. In M' , parts of the background near the contour of the relief are lifted while relief regions near the contour are depressed (in the case of raised reliefs, vice versa for lowered reliefs). Although M' is visually similar to the base surface obtained

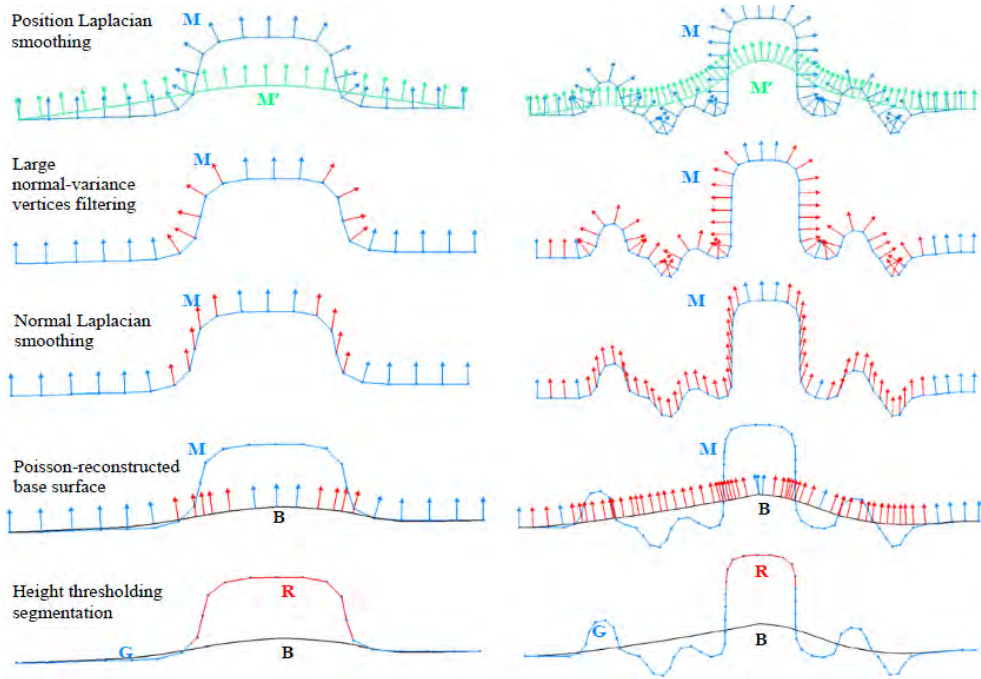


Figure 2: Relief extraction. (Top) By ignoring vertices whose normals change significantly before and after position Laplacian smoothing, reliable normals (blue normals in the second row) are selected as constraints in Laplacian smoothing of the normal field (center). The base surface B is reconstructed by solving a Poisson equation with boundary constraints (fourth row), allowing the relief R to be separated from the background G by height thresholding.

by Poisson smoothing, they may lead to large differences in the extracted reliefs, since reliefs take the form of low-height details. Figure 3 shows the poor results that would be obtained by directly using M' , and compares them to the actual results obtained by our full algorithm.

We now consider each step in detail.



Figure 3: Duck relief extraction result using our full algorithm (middle) and simply using a position Laplacian smoothed surface (right) as the base surface

3.1. Normal smoothing

To perform normal smoothing, we first use weighted Laplacian smoothing to smooth the mesh M , preserving the outer boundary. To do so, we minimize the following energy function:

$$E_p = \|LV'\|^2 + \sum_{i=1}^{N_V} \alpha_i \|\mathbf{v}_i - \mathbf{v}'_i\|^2. \quad (1)$$

Here, \mathbf{v}'_i is the smoothed mesh vertex corresponding to \mathbf{v}_i in M , and the vertex set V' comprises all \mathbf{v}'_i . L is the Laplacian matrix calculated from Laplacian coordinates using uniform weights. We also tried cotangent weights, but found that they provided similar or worse extraction results, while being more expensive to compute. Eqn. 1 is similar to the function used by [30], except that we use a different weight scheme α_i . To take into account variations in normals, α_i is a weight measuring local variance of normals in the one-ring neighborhood $\mathcal{N}(\mathbf{v}_i)$ of \mathbf{v}_i , defined by

$$\alpha_i = 1 / \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} \|\mathbf{n}_j - \mathbf{n}_i\|^2 \quad (2)$$

at all vertices, except at boundary vertices where it is set to 10^8 .

The first energy term measures the smoothing of base surface. The second term provides weights according to normal variation, while at the same time ensuring the boundary is fixed. A least-squares solution to this overdetermined linear system is obtained using Cholesky factorization to give the vertex coordinates of the smoothed surface. This smoothed surface is not suitable for use as a base surface, as it follows the general height of the relief, whereas the base surface should reflect the height of the original background. However, it provides a way of detecting vertices whose normals are reliable, allowing us to estimate a smooth normal field over the whole surface.

We detect these reliable vertices by comparing the original and smoothed (unit) normals, and mark as reliable those whose dot product exceeds a certain value. We use a fixed threshold of 0.99 which seems to work well in all cases we have tried—there is no need for manual setting of this threshold.

We then estimate the normals of the base mesh by smoothing the normals of M using uniform Laplacian smoothing, constrained by the reliable normals N_S . We use a function similar to Equation 1, where position variables are now replaced by normals: we minimize the following energy:

$$E_n = \|\mathbf{L}\mathbf{N}'\|^2 + \sum_{i=1}^{N_S} w \|\mathbf{n}_i - \mathbf{n}'_i\|^2, \quad (3)$$

where the weight w is set to 10^6 . These smoothed normal vectors are used as input to Poisson surface reconstruction in the next stage.

3.2. Base surface reconstruction

A good estimated base surface should be close to the real base surface. Intuitively, the problem can be seen as one of collapsing protruding relief regions back onto the base. To do so, we use a Poisson-based gradient domain

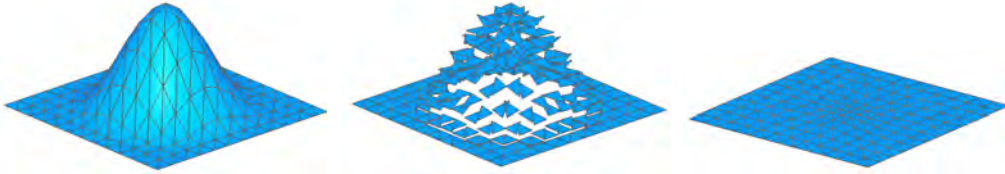


Figure 4: Base surface reconstruction. For the input mesh (left), each triangle is locally reoriented to agree with the smoothed normal field. The triangles become disconnected (center). The Poisson equation stitches the triangles together again in new positions, giving the base surface underlying the bumpyplane (right).

technique [17], in which the outer vertices are used as fixed vertex constraints (this assumes that the relief does not extend to the boundary).

In detail, for each triangle t_i with vertices $(\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c)$, we compute a local rotation matrix \mathbf{R}_i to bring its normal into alignment with the smoothed normal. The rotation matrix \mathbf{R}_i is defined by a rotation axis \mathbf{u} , given by the unit cross-product of the two normals, and the angle θ of rotation around \mathbf{u} needed to align the two normals.

Applying these local rotations separately to each mesh triangle would cause them to become disconnected. Constraining corresponding vertices of adjacent triangles to agree leads to a Poisson system in which the altered normals act as a guidance field. Solving it, with boundary conditions to keep the outer part of the mesh belonging to the background unchanged, gives the desired smooth base surface. Figure 4 illustrates the concept of base surface reconstruction using the smoothed normal field.

3.3. Thresholding for relief extraction

Having estimated the base surface, we compute the height of each vertex relative to it by using a simple moving-least-squares projection algorithm [31].

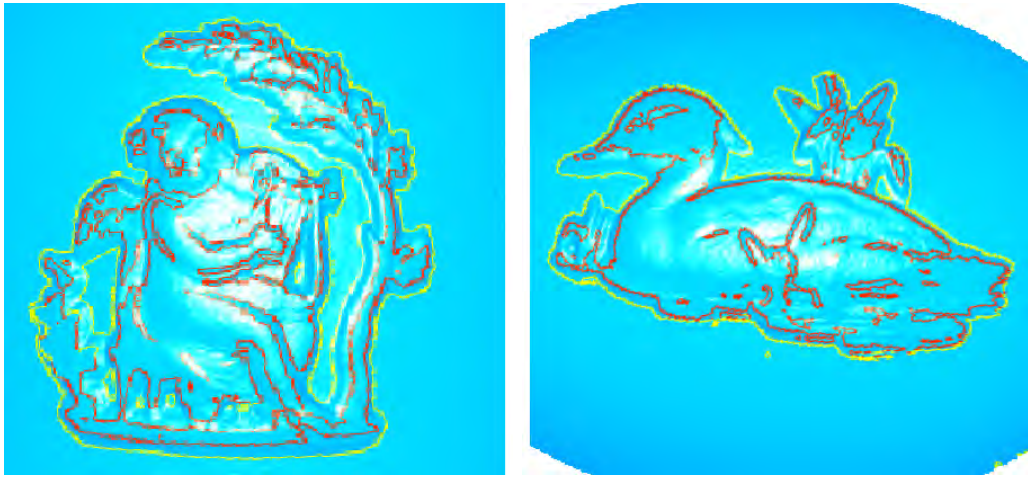


Figure 5: Upper (red) and lower (yellow) extraction boundaries.

We then use a Gaussian mixture model (GMM) on the distribution of the height values [5] to segment the relief R from the background G . In practice, a relief typically has two natural threshold boundaries corresponding to a lower value where the relief starts to rise from the background, and a higher inner one where the relief tends to flatten out (see Figure 5). The intersection of the GMM is used as the threshold for segmentation.

3.4. Experiments and discussion

Figures 6 and 7 show reliefs extracted from porcelain and lacquerware respectively. Our results are comparable to those of [2, 3, 5], and at times show some improvements. A further advantage of our algorithm is that it is fully automatic without the requirement for the user to initialize a snake [2, 3], or to determine parameters for base surface estimation [5]. Like [5], our algorithm can determine background holes within the relief, while Liu’s snake methods needs further background surface estimation steps to do so [2, 3].



Figure 6: Left to right: duck reliefs extracted by methods in [2] and [5], and by our approach. Highlighted artifacts in the first two are absent from our result.



Figure 7: Lacquerware relief extraction using the method in [3] (left, the red curve is the boundary), the method in [5] (middle), and our method (right).

Our method at times avoids artifacts produced by these earlier methods, leading to more precise results—see Figure 6, in which subtle but important differences can be seen. Figure 7 shows that our approach also can handle reliefs with textured backgrounds with reasonable success, and again, gives a cleaner result.

As our approach relies on solving a *sparse* linear system, relief extraction is fast. A model with $60K$ vertices takes less than 2 seconds to process, while the method in [5] requires about 16 seconds to achieve similar results, and the methods in [2, 3] take even longer.

4. Relief editing

In addition to being able to extract reliefs, we provide a range of relief editing tools to meet the needs of designers, again using differential coordinates. These tools provide both global transformations (translation, rotation, and scaling) of the whole relief, as well as a local deformation tool for detail

modification (see Figure 8). In each editing process, a Poisson-based composition step is used to merge the edited relief with the background G . The same basic approach can be used to transfer the relief onto a new object after editing, using the automatic correspondence establishment approach in [17].

4.1. Global transformations

Our global transformation tools allow the user to move the relief to another location on the background, to re-orient it, or to resize it. We first explain how translation is performed; the other transformations follow a similar approach.

4.1.1. Translation

Pseudocode for our translation algorithm is given in Algorithm 1, while the translation mechanism is illustrated in Figure 9. The user selects one point \mathbf{v}_h of the relief as a handle, and drags it arbitrarily across the surface to a new position. We track the handle as it moves, and project the path onto the base surface by simple use of the MLS algorithm [31]. We divide this path into small equal length polygonal segments on the surface, which are used to incrementally move the relief across the surface (we use a segment length equal to half the mean mesh edge length). During each incremental step, we calculate the normal \mathbf{n}_h of the projected handle at the starting position, and the associated local displacement $\Delta\mathbf{d}_j$ between the start and end positions. The position of each vertex \mathbf{v}_i of the relief is updated by the *UpdatePosition* function. We firstly project each vertex onto the base surface, and calculate the normal \mathbf{n}_i of that point. The two outward normals are aligned by rotating \mathbf{n}_h to \mathbf{n}_i ; if this would imply a rotation larger than

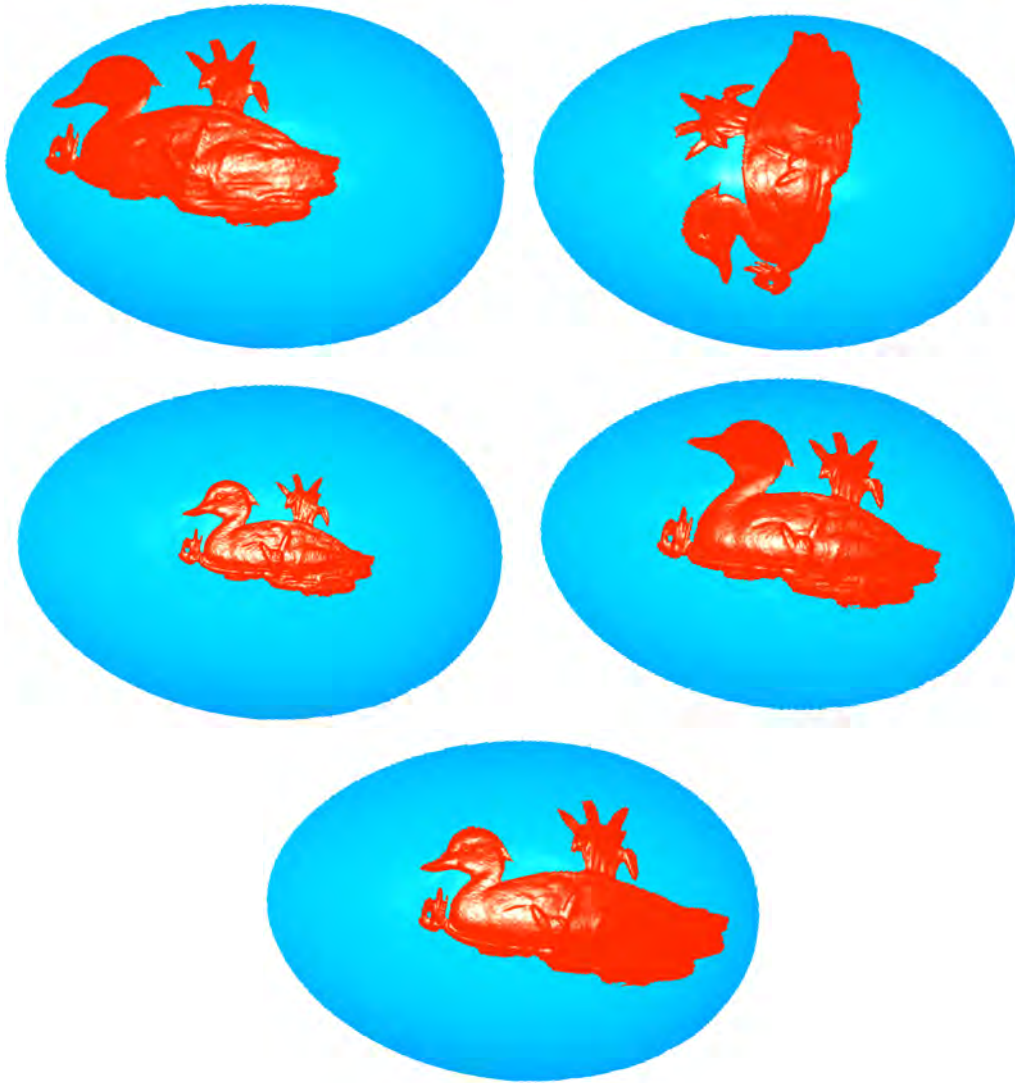


Figure 8: Editing the duck relief: translation, rotation, scaling, locally stretching the neck, stretching the rear of the duck after global scaling.

90° , we reverse the orientation of \mathbf{n}_i . This rotation also rotates the associated $\Delta\mathbf{d}_j$. We use its component perpendicular to \mathbf{n}_i as the direction in which to move \mathbf{v}_i , through a distance $\|\Delta\mathbf{d}_j\|$ to give a preliminary new vertex position.

Algorithm 1 Relief translation

- 1: Subdivide the projected handle trace
 - 2: **for** each incremental step j **do**
 - 3: Calculate normal of projected handle \mathbf{n}_h
 - 4: Calculate projected handle displacement $\Delta \mathbf{d}_j$
 - 5: **for** each vertex \mathbf{v}_i in R **do**
 - 6: UpdatePosition($\mathbf{v}_i, \Delta \mathbf{d}_j, \mathbf{n}_h$);
 - 7: **for** ($k = 0; k \leq \text{maxIterations}; k++$) **do**
 - 8: EdgeLengthPreservation(R);
 - 9: **for** each vertex \mathbf{v}_i in R **do**
 - 10: UpdateHeight(\mathbf{v}_i);
 - 11: PoissonComposition(R, G);
-

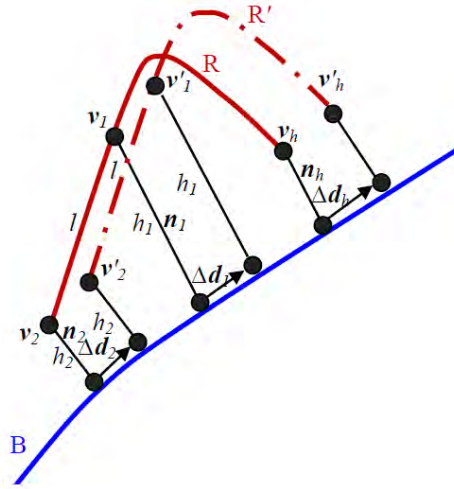


Figure 9: During global translation of a relief, as the user drags the handle \mathbf{v}_h , the height h_i of each vertex \mathbf{v}_i above the base surface should be preserved, and so should the length $l_{i,i+1}$ of the edge between vertices \mathbf{v}_i and \mathbf{v}_{i+1} .

We next update these preliminary new positions to bring the relief into close agreement with its original shape, by initially adjusting the *lengths* of mesh edges in the relief, and then *heights* of relief vertices above the background. To do the former, we use the *EdgeLengthPreservation* function, which iteratively adjusts vertex positions to satisfy the desired constraint that each relief edge $(\mathbf{v}_1, \mathbf{v}_2)$ should have an unaltered *length* l . Constraints of the form

$$C(\mathbf{v}_1, \mathbf{v}_2) = \|\mathbf{v}_1 - \mathbf{v}_2\| - l, \quad (4)$$

each yield a non-linear equation whose derivatives with respect to vertex positions are

$$\nabla_{\mathbf{v}_1} C(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{w}, \quad \nabla_{\mathbf{v}_2} C(\mathbf{v}_1, \mathbf{v}_2) = -\mathbf{w}, \quad (5)$$

where $\mathbf{w} = (\mathbf{v}_1 - \mathbf{v}_2)/\|\mathbf{v}_1 - \mathbf{v}_2\|$. Thus, the corrections made to \mathbf{v}_1 and \mathbf{v}_2 are

$$\Delta\mathbf{v}_1 = -\frac{1}{2}(\|\mathbf{v}_1 - \mathbf{v}_2\| - l)\mathbf{w}, \quad \Delta\mathbf{v}_2 = \frac{1}{2}(\|\mathbf{v}_1 - \mathbf{v}_2\| - l)\mathbf{w}. \quad (6)$$

Following [29], we repeatedly apply each edge constraint in a Gauss-Seidel type fashion, i.e. to each edge independently in turn.

We adjust the *height* of translated relief vertices above the base surface to agree with those in the original relief using the function *UpdateHeight*. This is done by simply moving each vertex to the correct height, in the direction of the base surface normal. (While this in principle changes the edge lengths, the overall effect is small, as shown later in our experimental results).

The final step is to use function *PoissonComposition* to merge the translated relief with the background surface where they meet: the background surface serves to provide Poisson boundary conditions for the Poisson-based

mesh composition technique in [17]. Correspondences are automatically established by matching every vertex on the relief boundary to the nearest vertex on the background boundary. The second boundary of the underlying background is obtained by removing the triangles along the boundary curve of the translated relief.

4.1.2. Rotation

To perform a global rotation operation, the user first selects a vertex to act as the centre of rotation. Although the vertex could be any point on the relief, it is both more intuitive and results in reduced distortion if this point is near the center of the relief. The normal \mathbf{n}_c of the projection of this vertex (\mathbf{v}_c) on the base surface is treated as the rotation axis. Thus, let \mathbf{v}_c be the center of rotation, and the tangent plane p_r of \mathbf{v}_c be the rotation plane. The user then selects a second vertex \mathbf{v}_h as a handle, and rotates it around the axis \mathbf{n}_c . We calculate a global rotation matrix \mathbf{R} from \mathbf{n}_c and the rotation angle of the projection of \mathbf{v}_h on the plane p_r . Then, \mathbf{R} is incrementally applied to each vertex \mathbf{v}_i in the relief. The trace of the path of \mathbf{v}_i projected on the base surface is again divided into small segments of equal length, and a similar process to that given in Section 4.1.1 is applied to the \mathbf{v}_i . In this case, however, each \mathbf{v}_i has its own trace.

4.1.3. Scaling

For global scaling, the user selects a vertex as an origin; again for best results this should be near the center of the relief. Let its projection point on the base be \mathbf{v}_s , and the tangent plane there be the scaling plane p_s . Another vertex is chosen as the scaling handle \mathbf{v}_h . We then project \mathbf{v}_h and all vertices

of the relief onto the plane p_s . As the user drags the handle, the scaling implied by the handle is computed on p_s , as the fraction relating its distance to \mathbf{v}_s and the distance between the projection of \mathbf{v}_h and \mathbf{v}_s . Scaling is then applied individually to each projected vertex outward from the center. We then obtain the trace of each projected vertex on the base surface, and again update the relief as before.

4.2. Local deformations

To perform non-rigid local deformation of the relief, we use a Poisson-based editing approach. Unlike the original Poisson-based editing algorithm [17], we use harmonic fields [32] to propagate user specified transformations at user chosen deformation handles to the remaining vertices, to update the gradient field. The geodesic distance field is used in [17], but is not in general smooth, and as a result, transformations for highly-protruding features of the mesh are attenuated. Harmonic propagation uses a smooth harmonic field on the mesh to overcome this problem. Specifically, we compute a harmonic scalar field on the relief using the formula $Ls = 0$, where L is the Laplacian matrix computed from the relief, and s is the desired scalar field. Here, we use the cotangent Laplacian operator, i.e., $w_{ij} = (\cot \alpha + \cot \beta)/2$, where α and β are angles opposite the edge in the two triangles that share edge e_{ij} . (For boundary edges, $w_{ij} = (\cot \alpha)/2$). This harmonic equation is solved with Dirichlet boundary conditions, which require that $s_i = 1$ for handles which have moved (the source of the propagation) and $s_i = 0$ for fixed handles (the sink of the propagation). We follow [32] in solving this problem.

Again, to edit the extracted relief, we project traces of handles representing each deforming vertex on the base surface, and proceed as before.

4.3. Results

We present various editing results in this section. Our system supports both global transformations: translation, rotation, and scaling of the whole relief, and local deformation of detail. Figures 1, 8, and 10 show various examples. Our scheme is robust, in the sense that in both global transformation and local editing, the results are pleasing, as distortion is avoided. While earlier work such as [5] provides a cut-and-paste operation that can glue reliefs to other objects, our editing tools go further in providing practical tools that let the user directly manipulate the reliefs over the background surface. Furthermore, as shown in Figure 8 (bottom), global translation and local editing can be combined to modify the relief. In our editing approach, handle manipulation is a direct yet powerful means to control the shape of a relief via user-controlled motions of a few vertices. This gives the user a uniform approach for all of our editing tools, which are simple to understand and use. A further advantage of our approach is that editing can be performed at interactive rates once the relief has been extracted, allowing the user to see the effects of his changes in real time.

It is difficult to provide a quantitative measure of success, however. Distortion is a tricky thing to measure, when reliefs are applied to different base surfaces. Furthermore, human vision and understanding are involved, simple geometric measures do not adequately capture the *perceived* quality of the results. Nevertheless, we believe our results to be of an acceptable standard for aesthetic use, and for completeness we summarize the distortion mea-

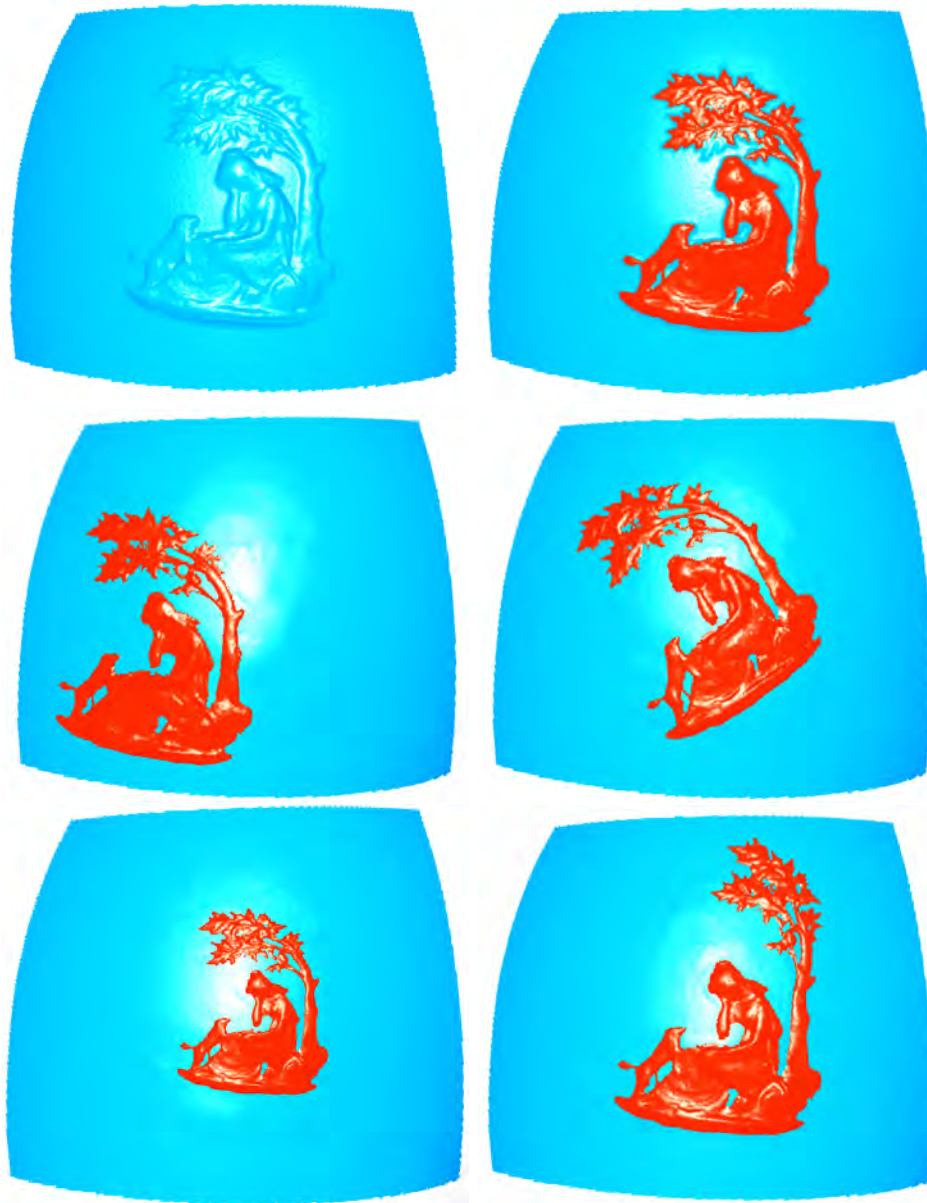


Figure 10: Relief extraction and editing. From left to right, top to bottom: input model, extracted relief, global translation, global rotation, global scaling, and local shape modification.

Table 1: Relative distortion error produced by editing. For global editing (translation, rotation, and scaling), SD error is standard deviation of errors in edge length ratios (before and after editing), while local editing error is measured by the root-mean-square (RMS) angular errors.

Example	SD error			RMS error
	Translation	Rotation	Scaling	Local Editing
Fig. 1	0.023	—	0.014	2.3°
Fig. 6	0.037	0.029	0.009	1.8°
Fig. 7	0.043	0.018	0.026	1.3°

surements for several editing results illustrated in this paper in Table 1: they demonstrate that the relative distortion is low. For global editing (translation, rotation, and scaling), we measured edge length ratios i.e. new edge lengths after editing divided by original lengths, as a proxy for distortion, and computed their standard deviation (taking any scaling into account, of course). For local edits, we instead computed the root-mean-square angular error for each triangle to assess its distortion, as edge lengths are intended to vary in this case.

5. Conclusions

Overall, we have provided a set of relief extraction and editing tools which meet real industrial requirements in a reverse engineering context. Relief extraction is fully automatic and rapidly obtains accurate results on real scanned reliefs, with fewer artifacts than previous approaches. Our relief

editing tools are the first specifically designed to manipulate reliefs on underlying base surfaces, as needed in the industrial relief design applications. Our editing results are plausible, and our editing tools are fast, simple, and easy to use. At the core of our methods is a differential-coordinates-based approach which has a solid theoretical foundation, yet is simple to implement.

In future, we will consider various limitations of the current approach. Certain types of reliefs cannot be expressed as a height function over a base surface—for example reliefs with undercuts. Reliefs with a similar height to a textured background are problematic to detect. Ideally we should take into account other constraints during relief editing, such as bending constraints [29], and the need to avoid collisions between topologically distant but geometrically close parts of the relief. More sophisticated distortion measures would be advantageous in assessing relief editing methods. Finally, a further useful operator would be one which could merge different reliefs to build a new composite relief.

References

- [1] S. Liu, R. R. Martin, F. C. Langbein, P. L. Rosin, Segmenting reliefs on triangle meshes, in: Proceedings of the 2006 ACM symposium on Solid and physical modeling, ACM, New York, NY, USA, 2006, pp. 7–16.
- [2] S. Liu, R. Martin, F. Langbein, P. Rosin, Background surface estimation for reverse engineering of reliefs, *International Journal of CAD/CAM* 7.
- [3] S. Liu, R. Martin, F. Langbein, P. Rosin, Segmenting geometric reliefs

- from textured background surfaces, *Computer-Aided Design and Applications* (2-3) (2007) 565–583.
- [4] S. Liu, R. Martin, F. Langbein, P. Rosin, Segmenting periodic reliefs on triangle meshes, in: *Mathematics of Surfaces XII*, 2007, pp. 290–306.
- [5] R. Zatzarinni, A. Tal, A. Shamir, Relief analysis and extraction, *ACM Transaction on Graphics (SIGGRAPH Asia)* 28 (5) (2009) Article No.: 136.
- [6] J. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, 1995.
- [7] O. Sorkine, M. Botsch, Tutorial: Interactive shape modeling and deformation, *Eurographics* (2009).
- [8] M. Botsch, O. Sorkine, On linear variational surface deformation methods, *IEEE Transactions on Visualization and Computer Graphics* 14 (1) (2008) 213–230.
- [9] W. Xu, K. Zhou, Gradient domain mesh deformation - A survey, *Journal of Computer Science and Technology* 24 (1) (2009) 6–18.
- [10] S. Anderson, M. Levoy, Unwrapping and visualizing cuneiform tablets, *IEEE Computer Graphics and Applications* 22 (6) (2002) 82–88.
- [11] M. Alexa, Differential coordinates for local mesh morphing and deformation, *The Visual Computer* 19 (2-3) (2003) 105–114.

- [12] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, H.-P. Seidel, Differential coordinates for interactive mesh editing, in: Shape Modeling International, IEEE Computer Society, 2004, pp. 181–190.
- [13] O. Sorkine, D. Cohen-Or, M. Alexa, C. Rössl, H.-P. Seidel, Laplacian surface editing, in: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2004, pp. 179–188.
- [14] A. Nealen, O. Sorkine, M. Alexa, D. Cohen-Or, A sketch-based interface for detail-preserving mesh editing, ACM Transactions on Graphics 24 (3) (2005) 1142–1147.
- [15] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, H.-Y. Shum, Large mesh deformation using the volumetric graph laplacian, ACM Transactions on Graphics 24 (3) (2005) 496–503.
- [16] H. Fu, O. K.-C. Au, C.-L. Tai, Effective derivation of similarity transformation for implicit laplacian mesh editing, Computer Graphics Forum 26 (1) (2007) 34–45.
- [17] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, H.-Y. Shum, Mesh editing with Poisson-based gradient field manipulation, ACM Transactions on Graphics 23 (3) (2004) 644–651.
- [18] R. Zayer, C. Rossl, Z. Karni, H.-P. Seidel, Harmonic guidance for surface deformation, Computer Graphics Forum 24 (3) (2005) 601–609.
- [19] T. Popa, D. Julius, A. Sheffer, Material-aware mesh deformations, in: IEEE Conference on Shape Modeling and Applications, Eurographics Association, 2006, pp. 22–30.

- [20] R. W. Sumner, J. Popović, Deformation transfer for triangle meshes, *ACM Transactions on Graphics* 23 (3) (2004) 399–405.
- [21] D. Xu, H. Zhang, Q. Wang, H. Bao, Poisson shape interpolation, *Graphical Models* 68 (3) (2006) 268–281.
- [22] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, H.-Y. Shum, Subspace gradient domain mesh deformation, *ACM Transactions on Graphics* 25 (3) (2006) 1126–1134.
- [23] O. K.-C. Au, C.-L. Tai, L. Liu, H. Fu, Dual Laplacian editing for meshes, *IEEE Transactions on Visualization and Computer Graphics* 12 (3) (2006) 386–395.
- [24] W. Xu, K. Zhou, Y. Yu, Q. Tan, Q. Peng, B. Guo, Gradient domain editing of deforming mesh sequences, *ACM Transactions on Graphics* 26 (3) (2007) Article No.:84.
- [25] Y. Lipman, O. Sorkine, D. Levin, D. Cohen-Or, Linear rotation-invariant coordinates for meshes, *ACM Transactions on Graphics* 24 (3) (2005) 479–487.
- [26] O. K.-C. Au, H. Fu, C.-L. Tai, D. Cohen-Or, Handle-aware isolines for scalable shape editing, *ACM Transactions on Graphics* 26 (3) (2007) 83.
- [27] Y.-K. Lai, S.-M. Hu, D. X. Gu, R. R. Martin, Geometric texture synthesis and transfer via geometry images, in: *ACM symposium on Solid and physical modeling*, ACM, New York, NY, USA, 2005, pp. 15–26.

- [28] H. Biermann, I. Martin, F. Bernardini, D. Zorin, Cut-and-paste editing of multiresolution surfaces, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, SIGGRAPH '02, ACM, New York, NY, USA, 2002, pp. 312–321.
- [29] M. Müller, B. Heidelberger, M. Hennix, J. Ratcliff, Position based dynamics, *Journal of Visual Communication and Image Representation* 18 (2) (2007) 109–118.
- [30] A. Nealen, T. Igarashi, O. Sorkine, M. Alexa, Laplacian mesh optimization, in: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, GRAPHITE '06, ACM, New York, NY, USA, 2006, pp. 381–389.
- [31] M. Alexa, S. Rusinkiewicz, M. Alexa, A. Adamson, On normals and projection operators for surfaces defined by point sets, in: Eurographics Symposium on Point-Based Graphics, 2004, pp. 149–155.
- [32] K. Xu, H. Zhang, D. Cohen-Or, Y. Xiong, Dynamic harmonic fields for surface processing, *Computer Graphics* 33 (3) (2009) 391–398.