ORIGINAL ARTICLE

A framework for digital sunken relief generation based on 3D geometric models

Meili Wang · Jian Chang · Jens Kerber · Jian J. Zhang

Abstract Sunken relief is a special art form of sculpture whereby the depicted shapes are sunk into a given surface. This is traditionally created by laboriously carving materials such as stone. Sunken reliefs often utilize the engraved lines or strokes to strengthen the impressions of a 3D presence and to highlight the features which otherwise are unrevealed. In other types of relief, smooth surfaces and their shadows convey such information in a coherent manner. Existing methods for relief generation are focused on forming a smooth surface with a shallow depth which provides the presence of 3D figures. Such methods unfortunately do not help the art form of sunken reliefs as they omit the presence of feature lines. We propose a framework to produce sunken reliefs from a known 3D geometry, which transforms the 3D objects into three layers of input to incorporate the contour lines seamlessly with the smooth surfaces. The three input layers take the advantages of the geometric information and the visual cues to assist the relief generation. We have modified the existing techniques of line drawings and relief generation, and then combine them organically for this particular purpose.

Keywords Sunken relief · Line drawings · 3D modeling · Digital art

M. Wang · J. Chang · J.J. Zhang (✉)
The Media School, Bournemouth University, Poole, BH12 5BB, UK
e-mail: jzhang@bournemouth.ac.uk

J. Kerber
MPI Informatik, Saarland University, Saarbrücken, Germany

1 Introduction

Relief is a type of sculptured artwork that is carved into a plane/surface or created by removing the unwanted pieces of material [1]. There are three types of relief [2]: high relief, bas relief and sunken relief. Sunken reliefs depend largely upon the shape which do not protrude from the surface, but are carved into it. In ancient Egyptian times, it was commonly used to illustrate stories and show the greatness of the pharaohs (as shown in Fig. 1). It is also used for inscriptions and engraved gemstones. Nowadays, sunken reliefs are widely used to decorate the surfaces of furniture, walls, buildings, and jewelery.

To produce a sunken relief, the sculptor carves the relief within a deeply incised line. As a result, the contours are sunk below the surrounding surface while leaving the highest parts on the surface level [2]. Figure 1 shows that the feature lines are important for depicting the edges of the objects; such lines represent most visual features of a sculpture. Because lines are flexible and concise, they are

Fig. 1 Sunken relief of ancient Egypt

considered the best way to represent general and abstract figures. In addition, in order to generate a realistic sunken relief, the smooth height transition between edges is also essential. Both the lines and the height transition are combined together to emphasize the presence of a 3D object.

Lines play an important role in forming a sunken relief. The external contours are defined as the principal lines, which are determined by the shape of the object itself and the viewer's perception. In addition, there are several other linear features which may be present in a relief. These are usually referred to as the interior lines because they are all contained within the overall shapes bounded by the external contours. Edges formed due to the connection of planes/surfaces at an angle may be rendered sharply so that they become a linear feature. These internal feature lines are defined as the secondary lines in a real sunken relief. Last but not the least, there are lines which are not derived from the three-dimensional form of the relief, but are overlaid to highlight the primarily graphic features [2].

Most existing research on relief generation is concerned with creating a smooth relief surface from either image or geometry input. Such a surface has a shallow depth but can deceive the human perception by creating a fake 3D shape with a much greater depth ratio. Few works have paid attention to the importance of lines. Our work introduces a novel framework that uniquely includes feature lines into relief creation, which distinguishes our work from its predecessors.

In our framework, we start from a full scale geometric mesh of a 3D object. In consequence, the user can benefit from abundant 3D models available either online or from laser scanning. It is more effective to operate in the image space than on the geometric mesh. Therefore, we transform the information conveyed by the mesh into three different images as three input layers in our framework: a line drawing image which provides input for contour lines; a rendered Lambertian image which shares the same light direction of the relief and provides visual cues; and a depth image which conveys the height information. By matching the target relief surface with the three given input images by an optimization process, we are able to maximize the presence of features in a sunken relief.

Our work has the following contributions:

- a novel framework to combine multiple channels of inputs for relief generation;
- incorporation of lines into smooth relief surface carving;
- combination of visual cues and real geometric information into the created relief.

Previous works use a single input only. They focus on either the visual cues by transforming an image into a relief or the geometric shape by compressing the geometry into a plane. Using multiple input channels, our method offers more detail and features consequently permitting further art stylization.

The paper is structured as follows: Sect. 2 reviews the relief generation techniques and line drawings from 3D models. Section 3 outlines the structure of our algorithm. Section 4 discusses how to prepare inputs from a 3D object for sunken relief generation. Section 5 derives the mathematical formulation for height optimization from multiple inputs. Section 6 demonstrates some results and Sect. 7 concludes the paper.

2 Related work

2.1 Digital relief generation techniques

Digital relief generation techniques are generally classified as reliefs from images, direct 3D sculpture systems and shape transformation [3]. Digital image processing can generate a relief style image via direct operations on pixels. However, the results of image processing give erroneous three-dimensional information that does not actually convey real height data.

In 2001, Perry and Frisken [4] developed a digital character sculpting system named Kizamu. It is mainly based on adaptively sampling distance fields (ADFs). It also incorporates a blend of new algorithms representing significant technical advances in modeling, introducing novel user interactions. Sourin [5] developed methods using programmed functions to represent shapes and manipulations. These function-based digital sculpting systems can produce carved-in sunken reliefs. Pasko et al. [6] also proposed a method to carve reliefs procedurally using the function representation of geometric objects. These digital systems create reliefs from scratch, which make them more comparable to the geometric modeling tools.

Manuel et al. [7] extended texture mapping techniques to fake the representation of 3D surface details visually, which on the contrary did not alter the geometric information at all. Other techniques, like bump mapping and displacement mapping can also create similar effects.

It is laborious and time-consuming to generate reliefs using 3D modeling software and sculpting system. To improve the efficiency, a few methods have been developed to generate reliefs that convert existing 3D models into relief models. In 1997, Cignoni et al. [8] developed a computer-assisted relief modeling system for generating 3D bas and high reliefs from 3D surface models. Later on, Belhumeur et al. [9] investigated the ambiguity of bas-relief generation with respect to surface reconstruction. It is reported that a linear compression of the 3D geometry can produce relief ambiguity but such implementation often suffers from loss of subtle details when the geometry is complicated. Research

in [10, 11] addressed the problem as geometry processing compared to the HDR (High dynamic range imaging [12]) or tone-mapping image compression. Weyrich et al. [13] introduced a perceptually motivated compression algorithm for relief generation. Later, Kerber et al. [14] improved their previous bas-relief generation work by adopting a bilateral filter to achieve good compression ratios without compromising the quality of surface details. This work was further enhanced to support artists and enthusiasts in creating bas-relief artworks from virtual scenes [15]. Sun et al. [16] also developed an automatic system to generate bas reliefs based on adaptive histogram equalization (AHE). All the aforementioned methods in this paragraph are related to flattening the models into a scaled height field with limited depth.

2.2 Shape-from-shading

Shape-from-shading, an approach to reconstructing the original 3D shape from a single image has been developed [17–19]. Shape-from-shading may be used for producing high reliefs but it requires a lot of user intervention. Wu et al. [20], using an altered shape-from-shading method, presented an interactive system for reconstructing surface normal from a single image. Based on this work, Wang et al. [21] developed a method to form a bas-relief surface from an image input.

Alexa and Matusik [22] proposed a method to generate relief surfaces automatically that could produce bas various images under different directional illumination. Although their method had an intrinsic link to shape from shading [17, 18]; the introduced discrete model with necessary degrees of freedom could overcome the theoretical limitations in shape from shading. The method in [22] can take multiple images as input as we do in this paper, however it focused on the illumination conditions while our approach is designed for including contour lines with other visual cues into a sunken relief.

2.3 Lines in sunken relief and 3D line drawings

Line drawings refer to extracting lines (the most obvious features of the model) from 3D models. The extracted lines could be applied to artistic stylization and abstraction work with applications ranging from illustrations to cartoons and gaming scenes.

Our work is inspired by the line drawings from 3D models since sunken reliefs depend considerably upon the carved lines in conveying the surface shape. Lines can convey other different things, including various combinations of lighting, materials, surface markings, and discontinuities [23].

The concept of contours and its connection to human perception was explained in [24]. Such contours in drawings are analogous to the principle lines in sunken reliefs. Later

on, creases were used, which was regarded as a big improvement for line drawings from 3D objects [25–27]. Creases are sharp features defined on the mesh surface. Suggestive contours [28–30], which are analogous to the interior lines in sunken reliefs, are minor on-surface features similar to contours that are view dependent and help to denote the local shape. Other drawing elements, such as hatching, are prevalent in more sophisticated line drawings. Many small lines can be combined in such a way that they simultaneously stylize the tone and material [31, 32]. For a more detailed review on line drawings, we refer to [23].

Algorithms for 3D line drawings can extract exactly the lines we need in generating the edges of a sunken relief. However, transforming them into a 3D carving in conjunction with smooth height transition is an unsolved problem. Current relief related research pays little attention to sunken relief generation, and it focuses on the recovery or creation of the height information. Our work, on the other hand, combines the line drawings input with height generation that visually highlights the shape feature on the generated sunken reliefs. To our knowledge, this is the first attempt to implement a method designed for sunken relief generation.

3 Outline of method

Our work aims to generate sunken reliefs from a known 3D object. Most important to note, we will overlay feature lines on the generated reliefs, thereby further enhancing the visual information conveyed to the viewer.

A 3D surface carries more information than a single image does [22]. In our implementation, we start from a triangular mesh by translating it into multiple inputs: a picture of plain line drawings, an image rendered using Lambertian shading, and a depth image. These inputs are combined together in a composition procedure to produce the final relief art piece. These procedures are illustrated in Fig. 2. The input of line drawings helps to create feature lines in
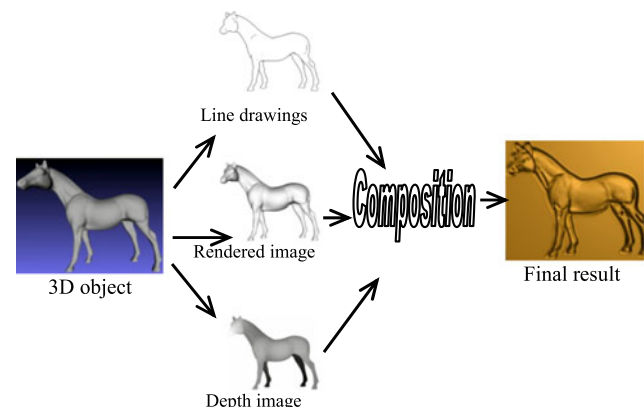

Fig. 2 Procedures of the proposed method

the sunken relief at the desired places. The other two inputs work in a conjugate manner to maintain a smooth relief surface.

4 Input pre-processing

4.1 Line drawings Input

There exists a large body of literature addressing the problem of extracting line drawings from exiting 3D models [25, 28, 31]. Many works contribute to the generation of different stylization and content related drawing, which is beyond the scope of our discussion. As the lines in the sunken relief highlight the contours, we only need to extract the plain line drawings. Figure 3 shows an example of the line drawings we created from a 3D mesh. Two types of line are included in our line drawings: contours (occluding contours and silhouettes) and suggestive contours. We use an object-space algorithm developed in [28] to extract the line drawings. A brief summary of the algorithm is shown below.

Contours Given a smooth surface, the contour lines are defined as a set of points that lie on the surface and whose surface normal is perpendicular to the view vector:

$$\mathbf{n}(\mathbf{p}) \cdot \mathbf{v} = 0 \qquad (1)$$

$\mathbf{p}$ is a point on the surface, $\mathbf{n}$ is its normal and $\mathbf{v}$ is the view vector. This set of points form disconnected curves on the surface. By projecting the visible part of the lines onto the image plane, we produce a line drawing defined by the occluding contours and silhouettes.

Suggestive contours Suggestive contours relate to the ridges and valleys as inflection points from a particular view on the surface, which involve computation of the local extrema of the curvature. A direction vector $\mathbf{w}$ is defined as the projection of view vector on the tangent plane at point $\mathbf{p}$.

$$\mathbf{w} = \frac{\mathbf{v} - (\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}) \cdot \mathbf{n}(\mathbf{p})}{\|\mathbf{v} - (\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}) \cdot \mathbf{n}(\mathbf{p})\|} \qquad (2)$$

It is noted that the local minima of $\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}$ in direction $\mathbf{w}$ suggests a possible location on a ridge/valley. This local minima corresponds to the zero value of the directional derivative of $\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}$ along direction $\mathbf{w}$ at point $\mathbf{p}$ where the second-order directional derivative along $\mathbf{w}$ is positive:

$$\nabla_{\mathbf{w}}(\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}) = 0, \quad \text{and}$$
$$\nabla_{\mathbf{w}}\nabla_{\mathbf{w}}(\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}) > 0 \qquad (3)$$

where $\nabla_{\mathbf{w}}$ denotes the directional derivative along $\mathbf{w}$. This condition is also equivalent to the zero value of radial curvature κ_r along direction $\mathbf{w}$ where the directional derivative along $\mathbf{w}$ of the radial curvature is positive:

$$\kappa_r = 0, \quad \text{and}$$
$$\nabla_{\mathbf{w}}\kappa_r > 0 \qquad (4)$$

The suggestive contours are generated by projecting those points on the surface that satisfy the aforementioned condition.

In our practice, we notice that direct usage of the line drawings generated from an arbitrary model may not be practical. Too many strokes make the relief appear bumpy, as some important geometric features are concealed or broken by redundant engraved lines. To overcome this issue, the user is provided with a tool to pre-process the mesh to control the input by filtering out the unwanted details. We choose the Laplacian smooth operator to smooth the mesh input. There are a number of different approximations for the Laplacian operator; each has its special usage. Hahan [33] describes its discrete approximation.

$$L(x_i) = \frac{1}{\sum_{w_{ij}}} \sum_{j \in N(i)} w_{ij}(x_j - x_i) \qquad (5)$$

where w_{ij} is the weight for edge defined by (x_i, x_j). There are several schemes to define the weights w_{ij}. Among those schemes, cotangent weight: $w_{ij} = \cot(\alpha_{i,j}) + \cot(\beta_{i,j})$, is widely used, where $\alpha_{i,j}$ and $\beta_{i,j}$ are the angles opposite to edge (x_i, x_j). An example of the practice of controlling the stroke density is shown in Figs. 3(b) and 3(c).

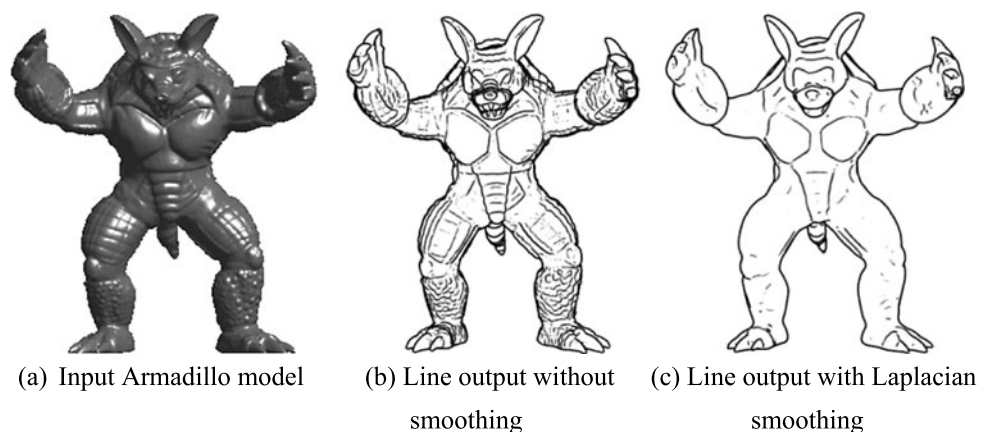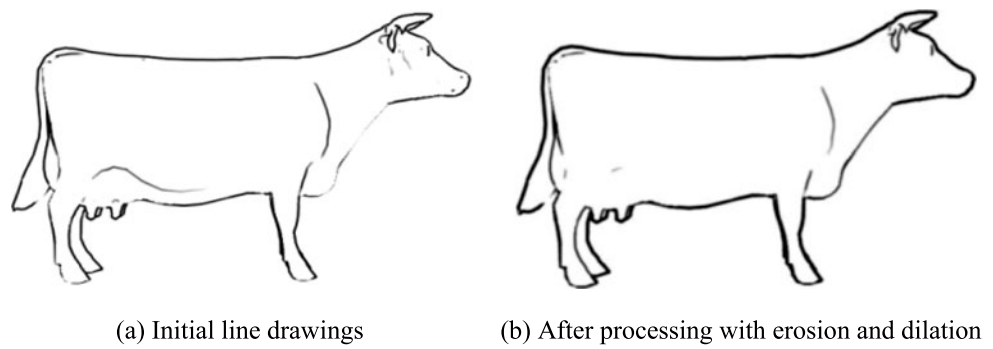**Fig. 3** An example of Armadillo model

(a) Input Armadillo model

(b) Line output without smoothing

(c) Line output with Laplacian smoothing

Fig. 4 Processing the extracted lines

(a) Initial line drawings | (b) After processing with erosion and dilation

Once the line drawings is produced, additional erosion and dilation operations that directly act on the pixels can be applied to tidy the image up by removing the noise and increasing the width of the lines making them more conspicuous (see Fig. 4).

4.2 Lambertian shading input

Images rendered with the Lambertian model are used as one input source to help recover the height information in the relief. Images rendered with the Lambertian model implicitly codes the surface normal at each point; it helps to recover the height information as the change of the surface normal provides essential information for the perception by the human eyes. The pixel value I is computed from the dot product of normal $\mathbf{n}$ and light direction $\mathbf{m}$ which point from the surface point toward the light source:

$$I = \alpha\big(\mathbf{n}(\mathbf{p}) \cdot \mathbf{m}(\mathbf{p})\big) \tag{6}$$

where α is the intensity of the light.

In our practice, we align the light source with the view point, which means we actually render $\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}$. In the production process, the light direction can align with the real setting for the relief display accordingly to allow the albedo of the generated relief in line with the geometry exactly.

4.3 Depth image input

It is beneficial to start from a 3D mesh as it provides the height information. Belhumeur et al. [9] suggested that a linear compression of the height could lead to ambiguity for some particular view points of the generated relief. Instead of using the simple linear compression approach, we use optimization approaches described in Sect. 5 to find a robust solution. Using the height information as input, we simply render its depth on the image projection plan as a gray image.

5 Relief height generation

To generate a relief, we initiate our inputs by translating a 3D object into three separate images: a line drawing, a Lam-
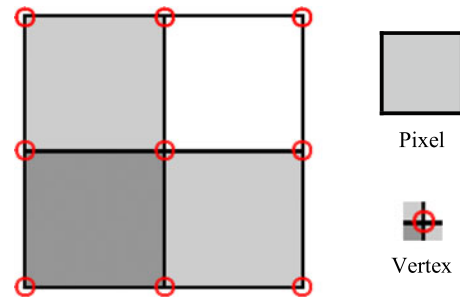

Fig. 5 The surface discretization of the relief in relation to pixels of an input image, where red circles represent the vertices' locations right on the corners of pixels

bertian shaded image, and a depth image. Those images are generated from the same view point and on the same projection plane to ensure that they are perfectly aligned with each other in terms of their x, y coordinates in the pixel domain. The image is set to have a size of m by n.

In a classic fashion, the discrete surface of a relief is created from an array of points that align with the pixels in the input image and their height is adjustable accordingly of generating a smooth shaded surface. Alexa and Matusik [22] suggested a new arrangement of using four connected triangles to calculate the value at a given pixel, where a pixel of the input image is modeled as a small pyramid in the relief. In our paper, we selected a staggered layout as shown in Fig. 5, where the actual relief grid for storing height data is selected by shifting the pixel grid in x and y directions for half a pixel size.

We define the pixel value of the relief grid node (the red circle in Fig. 5) at (x, y) by averaging the values of its four pixel neighbors,

$$I(x, y) = \frac{1}{4}\bigg(I\Big(x - \frac{1}{2}, y - \frac{1}{2}\Big) + I\Big(x - \frac{1}{2}, y + \frac{1}{2}\Big)$$
$$+ I\Big(x + \frac{1}{2}, y - \frac{1}{2}\Big) + I\Big(x + \frac{1}{2}, y + \frac{1}{2}\Big)\bigg) \tag{7}$$

we can define the gradient operation of the intensity at the given grid location (x, y) using the same four pixel values

as

$$\nabla I = \begin{pmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{pmatrix}$$

$$\frac{\partial I(x,y)}{\partial x} = \frac{1}{2}\left(I\left(x+\frac{1}{2}, y-\frac{1}{2}\right) + I\left(x+\frac{1}{2}, y+\frac{1}{2}\right)\right.$$
$$\left. - I\left(x-\frac{1}{2}, y-\frac{1}{2}\right) - I\left(x-\frac{1}{2}, y+\frac{1}{2}\right)\right)$$

$$\frac{\partial I(x,y)}{\partial y} = \frac{1}{2}\left(I\left(x-\frac{1}{2}, y+\frac{1}{2}\right) + I\left(x+\frac{1}{2}, y+\frac{1}{2}\right)\right.$$
$$\left. - I\left(x-\frac{1}{2}, y-\frac{1}{2}\right) - I\left(x+\frac{1}{2}, y-\frac{1}{2}\right)\right)$$

$$(8)$$

Alternatively, we can define the gradient operation of the height value at the pixel location in a similar compact fashion when we estimate the surface normal in Sect. 5.2. Such compact layout helps to increase the resolution for our reconstruction of the relief surface. In the following, we describe how to derive an independent energy function for each of the aforementioned inputs.

5.1 Depth input

With respect to the input depth image, it is possible to transform it into a piece of relief by a linear compression. Such direct operation may cause loss of local features and the resulting relief could look dull due to the lack of depth incorporated information. We process the height information with a non-linear compression by recovering the surface from computing the Poisson equations.

Let us denote the depth value at position (x, y) as $I_d(x, y)$. The second-order derivative value is computed by

$$g = \nabla \cdot G(\nabla I_d) \tag{9}$$

G is a non-linear high dynamic compression function [12], which takes the following form:

$$G(a) = \left(\text{sgn}(a)\frac{\gamma}{|a|}\left(\frac{|a|}{\gamma}\right)^{\beta}\right) \tag{10}$$

γ and β jointly control the compression ratio. We set $\gamma = 0.1$ and $\beta = 0.9$ in our experiments.

We then reconstruct the relief surface by solving the following Poisson equations:

$$\nabla^2 h(x, y) - g = 0 \tag{11}$$

Here $h(x, y)$ specifies the discrete height distribution. This Poisson equation can be numerically resolved. However, in order to integrate our solution with all three inputs, we define the energy terms and find the solution by minimizing the overall energy.

In this way, we have the energy term that relates to the depth image input as follows:

$$E_d = \sum_{x=2}^{m-1}\sum_{y=2}^{n-1}[h(x+1, y) + h(x-1, y) + h(x, y+1)$$
$$+ h(x, y-1) - 4h(x, y) - g(x, y)]^2 \tag{12}$$

5.2 Lambertian image input

Let us denote $\mathbf{L}$ as the light source. $\mathbf{L} = \alpha \mathbf{m}(\mathbf{p})$, where α is the light intensity and $\mathbf{m}$ is the light direction at point $\mathbf{p}$. The albedo (or the reflection radiance) at the point is

$$I = \mathbf{L} \cdot \mathbf{n} \tag{13}$$

with $\mathbf{n}$ as the surface normal at the point.

Using the staggered grid layout as shown in Fig. 5 to estimate the reflection radiance I of the relief at a pixel point $(x + 1/2, y + 1/2)$, we can use the four height values at the pixel's corners to compute the normal direction first. We can write the estimated normal as

$$\mathbf{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \frac{1}{n}\begin{pmatrix} -\frac{\partial h}{\partial x} \\ -\frac{\partial h}{\partial y} \\ 1 \end{pmatrix} \tag{14}$$

where

$$\frac{\partial h}{\partial x} = \frac{1}{2}\big(h(x+1, y) + h(x+1, y+1) - h(x, y)$$
$$- h(x, y+1)\big),$$
$$\frac{\partial h}{\partial y} = \frac{1}{2}\big(h(x, y+1) + h(x+1, y+1) - h(x, y)$$
$$- h(x+1, y)\big)$$

and

$$\hat{n} = \sqrt{1 + \left(\frac{\partial h}{\partial x}\right)^2 + \left(\frac{\partial h}{\partial y}\right)^2}$$

as the scale factor to normalize the normal vector. In the latter optimization steps, we need to linearize the system by estimating the scale factor with the results from the previous iteration rather than computing it at the current time. The scale factor is initiated as 1 when computing starts. Given $\mathbf{n}$ as the normal we can compute the reflection radiance pixel by pixel using (13).

The input Lambertian image has the intensity value at a given position $(x + 1/2, y + 1/2)$ as $I_L(x + 1/2, y + 1/2)$. Recalling our staggered grid setting, we note that both the estimated reflection radiance and the intensity image are defined at the pixel grids and their gradients are defined at the relief grid nodes by (8). Using the image gradient compression method [12], we can have the compressed image gradient of the input as $G(\nabla I_L)$, where the compression function

is defined in (10). We expect the gradient of the reflection radiance identical to the compressed input value, that is,

$$\nabla I(x, y) - G(\nabla I_L) = 0 \tag{15}$$

We therefore define the related energy term to the Lambertian image input as follows:

$$
\begin{aligned}
E_L = \sum_{x=1}^{m-1}\sum_{y=1}^{n-1}\Bigg(& \frac{1}{2}\Big(I\Big(x+\frac{1}{2}, y-\frac{1}{2}\Big) \\
&+ I\Big(x+\frac{1}{2}, y+\frac{1}{2}\Big) - I\Big(x-\frac{1}{2}, y-\frac{1}{2}\Big) \\
&- I\Big(x-\frac{1}{2}, y+\frac{1}{2}\Big)\Big) - G\Big(\frac{\partial I_L}{\partial x}\Big)\Bigg)^2 \\
+ \sum_{x=1}^{m-1}\sum_{y=1}^{n-1}\Bigg(& \frac{1}{2}\Big(I\Big(x+\frac{1}{2}, y+\frac{1}{2}\Big) \\
&+ I\Big(x-\frac{1}{2}, y+\frac{1}{2}\Big) - I\Big(x+\frac{1}{2}, y-\frac{1}{2}\Big) \\
&- I\Big(x-\frac{1}{2}, y-\frac{1}{2}\Big)\Big) - G\Big(\frac{\partial I_L}{\partial y}\Big)\Bigg)^2
\end{aligned}
\tag{16}
$$

5.3 Line drawings input

To generate the engraved line effect for a sunken relief, we include a separate line drawing input layer. We define another energy item specially associated to this input. Given the line drawings, we first initiate a stencil based on this. The stencil is an image whose strokes are one pixel wider than those in the input. We denote $I_w(x + 1/2, y + 1/2)$ as the pixel value at position $(x + 1/2, y + 1/2)$ of the line drawings input. The relief grid value $S(x, y)$ of the stencil template is defined as follows:

$$
S(x, y) = \begin{cases} 1 & \text{if } \sum_{i=x-\frac{1}{2}}^{x+\frac{1}{2}} \sum_{j=y-\frac{1}{2}}^{y+\frac{1}{2}} I_w(i, j) > 0; \\ 0 & \text{else} \end{cases}
\tag{17}
$$

The stencil value is zero only when all of its four neighboring pixels are zero.

Using stencil S, we can filter out the influence of those pixels not adjacent to a given stroke to avoid unnecessary computation. In order to engrave lines on the relief according to the line drawings input, we expect that the following relationship holds:

$$S \cdot \big(\nabla I(x, y) - G(\nabla I_w)\big) = 0 \tag{18}$$

I being the intensity of the reflection radiance calculated by (13) whose values are defined at the pixel grid and whose gradients are defined at the relief grid and G is the non-linear compress function of (10).

Therefore, we have the definition of the energy term for the line drawings input as,

$$
\begin{aligned}
E_w = \sum_{x=1}^{m-1}\sum_{y=1}^{n-1} S(x, y)\Bigg(& \frac{1}{2}\Big(I\Big(x+\frac{1}{2}, y-\frac{1}{2}\Big) \\
&+ I\Big(x+\frac{1}{2}, y+\frac{1}{2}\Big) - I\Big(x-\frac{1}{2}, y-\frac{1}{2}\Big) \\
&- I\Big(x-\frac{1}{2}, y+\frac{1}{2}\Big)\Big) - G\Big(\frac{\partial I_w}{\partial x}\Big)\Bigg)^2 \\
+ \sum_{x=1}^{m-1}\sum_{y=1}^{n-1} S(x, y)\Bigg(& \frac{1}{2}\Big(I\Big(x+\frac{1}{2}, y+\frac{1}{2}\Big) \\
&+ I\Big(x-\frac{1}{2}, y+\frac{1}{2}\Big) - I\Big(x+\frac{1}{2}, y-\frac{1}{2}\Big) \\
&- I\Big(x-\frac{1}{2}, y-\frac{1}{2}\Big)\Big) - G\Big(\frac{\partial I_w}{\partial y}\Big)\Bigg)^2
\end{aligned}
\tag{19}
$$

5.4 Compositing relief by minimizing the energy

To recover the height information, we minimize the superposition of all existing energy terms with respect to the height values at the corners of the pixels. The overall energy is written as follows:

$$E = w_d E_d + w_L E_L + w_w E_w \tag{20}$$

This illustrates that the weighting coefficients control the influence of each term. The gradient of the reflection radiance non-linearly depends on the height alternation. This dependency is linearized by estimating the scale factor which normalizes the normal vector by the results from previous optimization step.

The optimization problem is defined as,

$$
\begin{aligned}
\min_{h} \quad & E(h) \\
\text{s.t.} \quad & h < 0, \\
& -h < h_{\max}
\end{aligned}
\tag{21}
$$

The constraints of height ensure the relief is sunk into a given flat plane. $h_{\max}$ explicitly specifies the maximum depth that can be carved into the plane. We use the Matlab optimization Toolbox to solve this optimization problem.

6 Results

Our algorithm is capable of providing relief models as triangular meshes. Such meshes can be further simplified for less storage or to lower the computational cost for future operation. Other post processing steps, such as texturing and shading can be added to create further embellished results.
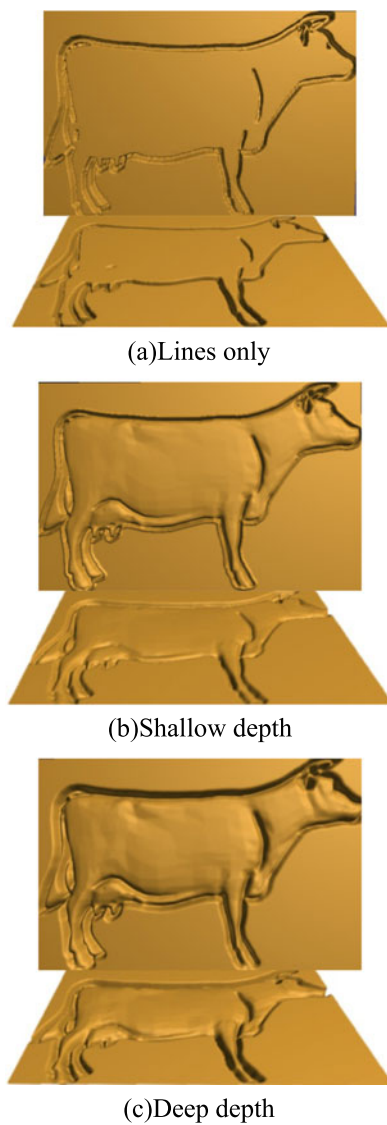
(a)Lines only

(b)Shallow depth

(c)Deep depth

Fig. 6 Comparison of reliefs with different inputs

(a) Initial lines input   (b) Laplacian smooth

Fig. 7 Comparison of reliefs with different line density

(a)   (b)

(c)   (d)

Fig. 8 Bas reliefs produced by the methods of (**a**) Cignoni et al. [8], (**b**) Kerber et al. [11], (**c**) Kerber [14], and (**d**) Sun et al. [16]

As shown in Fig. 6(a), a plain carved relief only includes the line drawings input. Figures 6(b) and 6(c) represent our compositing results with all inputs. They have different depths of carving. It is noted that when combining both lines and the surface together, the relief looks much more convincing.

Figure 7 illustrates the results from different line drawings inputs with and without Laplacian smoothing. We can see that redundant strokes of input make the result (Fig. 7(a)) appear bumpy and hide the main shape features at some point, which can be enhanced (see Fig. 7(b)) with a mesh smoothing process before extracting the counter lines when preparing the line drawings input.

Figure 8 demonstrates the generated bas reliefs with different approaches: (a) by Cignoni et al. [8], (b) Kerber et al. [11], (c) Kerber [14], and (d) Sun et al. [16]. One can see from Fig. 8(a) that the simple linear compression is not suf-
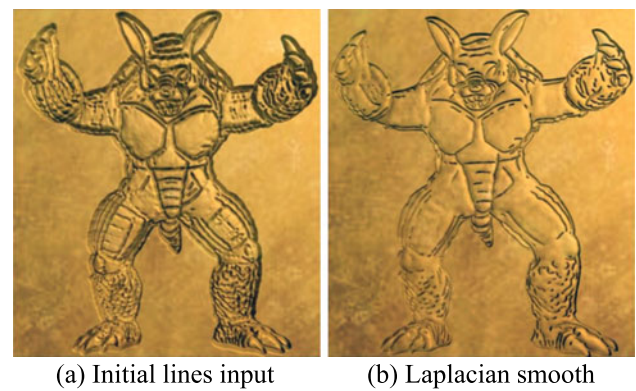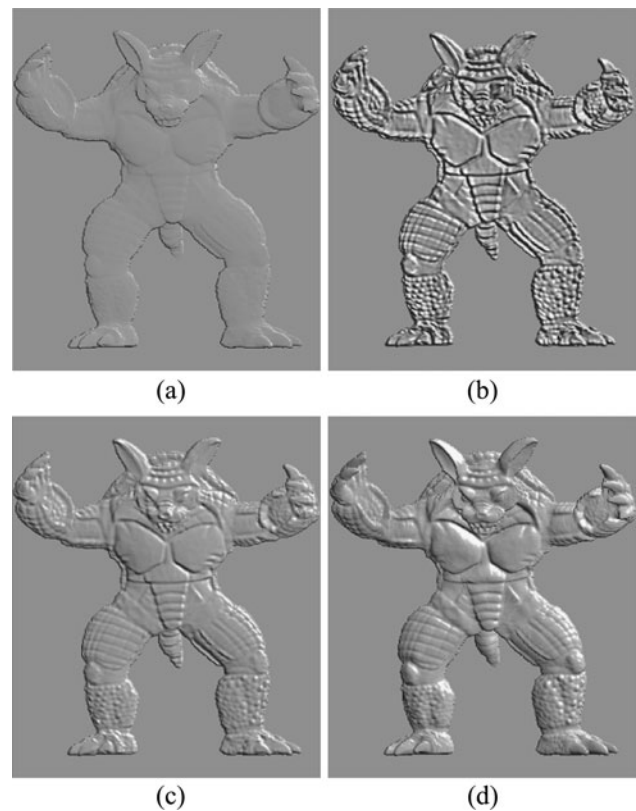
ficient for relief generation as many important features are missing. The others preserve the details well, which could be applied to generate sunken relief by simply shifting the height into its surface without highlight of counter lines. The results created by direct sinking a bas relief would be compromised and degenerated because the important feature lines with carved effects are missing, a factor upon which sunken relief mostly relies. The difference between our method (Fig. 7) and others (Fig. 8) is self-evident. Our approach benefits from using the contour lines, which impresses the presence of 3D features and addresses more vari-
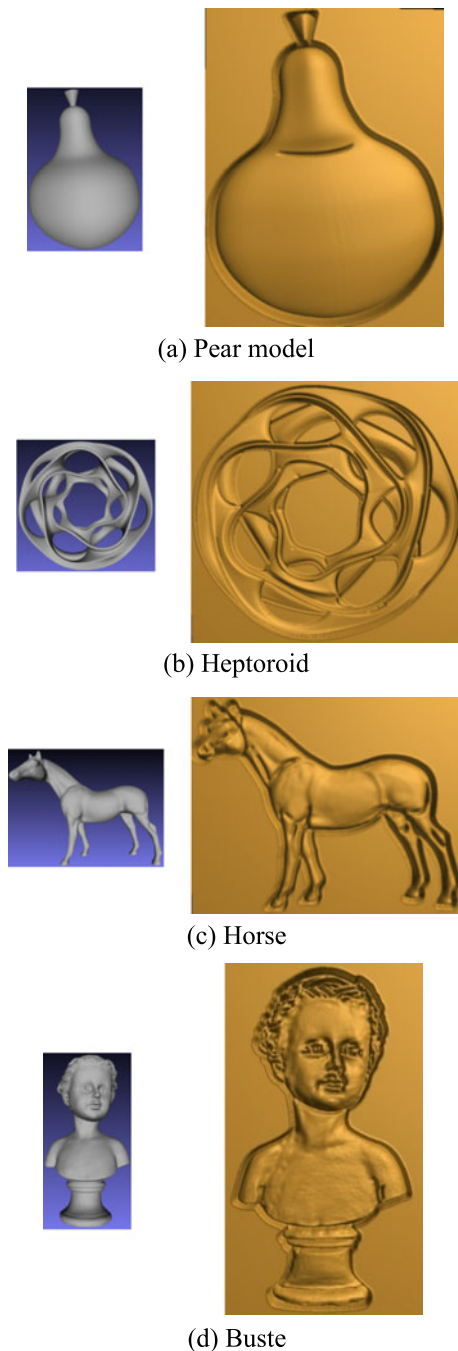
(a) Pear model

(b) Heptoroid

(c) Horse

(d) Buste

Fig. 9 Sunken relief results

ations in the art forms. Such lines are organically integrated with the smooth surfaces.

Figure 9 shows multiple results we have created with our method using different 3D models. It demonstrates that our method is robust in handling both simple abstraction forms (Fig. 9(a), the Pear), and complex geometric features (Fig. 9(d), the Buste). Very low frequent but smooth features like the muscles along the shoulders of the horse or the neck of the pear could hardly be preserved if only the height infor-

mation is taken into account. The example of the heptoroid shows that the depth order remains visible and correct.

In our experiment, different 3D meshes are used as inputs to create the sunken reliefs (Figs. 6, 7, 9). The user is able to control the sunken relief outputs with different settings in preparing their inputs. The PC we used is HP Workstation 4300, with Intel Pentium Dual CPU 3.2 GHz and 2 GB RAM, which is equipped with NVIDIA GeForce 7900 GTX video card. It takes about 6–20 seconds of processing time to create a relief using input images with resolution of 1024×768. For example, the cow model (Fig. 6(c)) was created with 6.67 seconds and Armadillo model (Fig. 7((b)) with 18.07 seconds. The variance of time depends on the resolution as well as the complexity of the reliefs. As we have implemented our algorithm in Matlab, we can foresee a potential improvement of performance when rewriting the code in C++.

7 Conclusion

In this paper, we have proposed a method to generate sunken reliefs using 3D models as input. Our experiments show that the combination of lines and height transition can effectively generate satisfactory results. Presented in the digital form, the generated 3D mesh for the resulting sunken relief can be used as input of design software for further refinement or can be directly used for machining a real piece of relief. Such relief form can also be used as references or prototypes for craftsmen in their actual carving practices.

Two major factors are involved in generating a sunken relief: feature edges and smooth height transition, prepared as separate inputs from a known 3D mesh. In the composition step, we minimize the overall energy of the related items to create the height field for smooth surfaces as well as incorporate the lines into a relief.

Although our method starts with a 3D object, the generation of the relief is based on image-like inputs, implying that our method can be directly adopted for relief generation from images. It is possible to extract feature lines and contours from a 2D image using exiting techniques, which makes our method feasible for sunken relief generation by including the images as additional input along with the original image.

Currently, our work is limited to creating sunken reliefs with certain line types (contours, creases etc.). It would be useful, but also challenging to stylize the relief with texture details to reflect the difference derived from the various carving tools used. Such work will be investigated in our future work.

References

1. Flaxman, J.: Lectures on Sculpture. J. Murray, London (1829)

2. Rogers, R.: Relief Sculpture. Oxford University Press, Oxford (1974)

3. Wang, M., Chang, J., Zhang, J.J.: A review of digital relief generation techniques. In: The 2nd International Conference on Computer Engineering and Technology (ICCET 2010), Apr. 2010, Chengdu, China (2010)

4. Perry, R.N., Frisken, S.F.: Kizamu: a system for sculpting digital characters. In: SIGGRAPH'01, pp. 47–56 (2001)

5. Sourin, A.: Functionally based virtual embossing. J. Vis. Comput. Animat. **17**(4), 258–271 (2001)

6. Pasko, A., Savchenko, V., Sourin, A.: Synthetic carving using implicit surface primitives. Comput. Aided Des. **33**(5), 379–388 (2001)

7. Oliveira, M.M., Bishop, G., Mcallister, D.: Relief texture mapping. In: SIGGRAPH'00, pp. 359–368 (2000)

8. Cignoni, P., Montani, C., Scopigno, R.: Computer-assisted generation of bas- and high-reliefs. J. Graphics Tools **2**(3), 15–28 (1997)

9. Belhumeur, P.N., Kriegman, D.J., Yuille, A.L.: The bas-relief ambiguity. Int. J. Comput. Vis. **35**(1), 33–44 (1999)

10. Song, W., Belyaev, A., Seidel, H.P.: Automatic generation of bas-relief from 3D shapes. In: SMI'07: Proceedings of the IEEE International Conference on Shape Modelling and Applications, pp. 211–214 (2007)

11. Kerber, J., Belyaev, A., Seidel, H.P.: Feature preserving depth compression of range images. In: Proceedings of the 23rd Spring Conference on Computer Graphics, pp. 110–114 (2007)

12. Fattal, R., Lischinski, D., Werman, M.: Gradient domain high dynamic range compression. ACM Trans. Graph. **21**(3), 249–256 (2002)

13. Weyrich, T., Deng, J., Barnes, C., Rusinkiewicz, S., Finkelstein, A.: Digital bas-relief from 3D scenes. ACM Trans. Graph. **26**(3), 32–39 (2007)

14. Kerber, J., Tevs, A., Belyaev, A., Zayer, R., Seidel, H.P.: Feature sensitive bas relief generation. In: IEEE International Conference on Shape Modelling and Applications, pp. 148–154 (2009)

15. Kerber, J., Tevs, A., Zayer, R., Belyaev, A., Seidel, P.: Real-time generation of digital bas-reliefs. Computer-Aided Des. Appl. **7**(4), 465–478 (2010)

16. Sun, X., Rosin, P.L., Martin, R.R., Langbein, F.C.: Bas-relief generation using adaptive histogram equalization. IEEE Trans. Vis. Comput. Graph. **15**(4), 642–653 (2009)

17. Agrawal, A., Raskar, R., Chellappa, R.: What is the range of surface reconstructions from a gradient field? In: Computer Vision—European Conference on Computer Vision, pp. 578–591 (2006)

18. Zeng, G., Matsushita, Y., Quan, L., Shum, H.Y.: Interactive shape from shading. In: Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 343–350 (2005)

19. Zhang, R., Tsai, P., Cryer, J.E., Shah, M.: Shape from shading: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **21**(8), 690–706 (1999)

20. Wu, T.P., Sun, J., Tang, C.K., Shum, H.Y.: Interactive normal reconstruction from a single image. In: SIGGRAPH Asia'08 (2008), Article 119, pp. 1–9

21. Wang, M., Chang, J., Pan, J., Zhang, J.J.: Image-based bas-relief generation with gradient operation. In: Proceedings of the 11th IASTED International Conference Computer Graphics and Imaging (CGIM 2010), Feb. 2010, Innsbruck, Austria (2010)

22. Alexa, M., Matusik, W.: Reliefs as images. ACM Trans. Graph. **29**(4), 1–7 (2010)

23. Rusinkiewicz, S., Cole, F., Decarlo, D., Finkelstein, A.: Line drawings from 3D models. In: SIGGRAPH 2008 Course Notes (2008). http://www.cs.princeton.edu/gfx/proj/sg08lines/

24. Koenderink, J.J.: What does the occluding contour tell us about solid shape? Perception **13**(3), 321–330 (1984)

25. Kalnins, R.D., Markosian, L., Meier, B.J., Kowalski, M.A., Lee, J.C., Davidson, P.L., Webb, M., Hughes, J.F., Finkelstein, A.: WYSIWYG NPR: drawing strokes directly on 3D models. ACM Trans. Graph. **21**(3), 755–762 (2002)

26. Markosian, L., Kowalski, M.A., Goldstein, D., Trychin, S.J., Hughes, J.F., Bourdev, L.D.: Real-time nonphotorealistic rendering. In: SIGGRAPH'97, pp. 415–420 (1997)

27. Raskar, R.: Hardware support for non-photorealistic rendering. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, pp. 41–47 (2001)

28. Decarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. ACM Trans. Graph. **22**(3), 848–855 (2003)

29. Decarlo, D., Finkelstein, A., Rusinkiewicz, S.: Interactive rendering of suggestive contours with temporal coherence. In: Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, pp. 15–145 (2004)

30. Sousa, M.C., Prusinkiewicz, P.: A few good lines: suggestive drawing of 3d models. Comput. Graph. Forum **22**(3), 381–390 (2003)

31. Cole, F., Golovinskiy, A., Limpaecher, A., Barros, H.S., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S.: Where do people draw lines? ACM Trans. Graph. **27**(3), 1–11 (2008)

32. Cole, F., Sanik, K., Decarlo, D., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S., Singh, M.: How well do line drawings depict shape? ACM Trans. Graph. **28**(3), 1–9 (2009)

33. Hahne, U.: Weighting in Laplacian Mesh Editing (2006). http://www.cg.tu-berlin.de/fileadmin/webdav/hahne/diploma/diploma.pdf

Meili Wang is a Ph.D. research student at the National Centre for Computer Animation, Bournemouth University. Her research interests include computer graphics, geometric modelling, image processing, visualization and virtual reality.

Jian Chang is a lecturer at the National Centre for Computer Animation, Bournemouth University. He received his Ph.D. degree in computer graphics in 2007 at the National Centre for Computer Animation, Bournemouth University. His research focuses on a number of topics relating to geometric modelling, algorithmic art, character rigging and skinning, deformation and physically-based animation. He also has strong interest in applications in medical visualization and simulation.

Jens Kerber obtained his Master's degree in Computer Science from Saarland University in 2007. After that he joined the Computer Graphics Department at the MPI Informatik in Saarbrücken as a Ph.D. candidate. In 2008 he moved on to the Statistical Geometry Processing Group within the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University. Besides from relief generation, his research interests include symmetry analysis in volume data.

Jian J. Zhang is Professor of Computer Graphics at the National Centre for Computer Animation, Bournemouth University. He leads the Computer Animation Research Centre. His research interests include computer graphics, computer animation, physically based simulation, geometric modelling, medical simulation and visualization.