Eurographics Symposium on Geometry Processing 2014
Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

Volume 33 (2014), Number 5

Real-time Bas-Relief Generation from Depth-and-Normal Maps on GPU

Zhongping Ji[1], Xianfang Sun[2], Shi Li[3], and Yigang Wang[3†]

[1] School of Computer Science, Hangzhou Dianzi University, China
[2] School of Computer Science and Informatics, Cardiff University, UK
[3] School of Media and Design, Hangzhou Dianzi University, China

Abstract

To design a bas-relief from a 3D scene is an inherently interactive task in many scenarios. The user normally needs to get instant feedback to select a proper viewpoint. However, current methods are too slow to facilitate this interaction. This paper proposes a two-scale bas-relief modeling method, which is computationally efficient and easy to produce different styles of bas-reliefs. The input 3D scene is first rendered into two textures, one recording the depth information and the other recording the normal information. The depth map is then compressed to produce a base surface with level-of-depth, and the normal map is used to extract local details with two different schemes. One scheme provides certain freedom to design bas-reliefs with different visual appearances, and the other provides a control over the level of detail. Finally, the local feature details are added into the base surface to produce the final result. Our approach allows for real-time computation due to its implementation on graphics hardware. Experiments with a wide range of 3D models and scenes show that our approach can effectively generate digital bas-reliefs in real time.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages

1. Introduction

Recently, the problem of automatic generation of bas-reliefs from 3D input scenes has received great attention. The key ingredient of this procedure is compressing the height field sampled from the input scenes with detailed geometric features preserved. The previous work has solved this problem to some extent. Most of them focused on designing sophisticated non-linear depth compression algorithms which induces very high computational cost. However in many scenarios, designing a bas-relief from a 3D scene is an inherently interactive task. Given a 3D scene, it requires tedious work for users to obtain aesthetically pleasing bas-reliefs. The user interactively selects a view angle of the 3D scene, as well as tweaks the appropriate parameter values by trial, then waits for the result. The computational cost of the algorithm which converts the 3D model to the bas-relief is the key factor to real-time bas-relief designing. In this paper, we design the converting procedure exploiting the parallel character of the modern graphics hardware, so that real-time design is made possible.

† Corresponding: wangyg@cad.zju.edu.cn

Contributions We develop a simple and fast modeling system for generating visually plausible bas-reliefs from 3D input scenes. We make three main improvements and contributions:

- We have developed a two-scale approach for converting 3D scenes to bas-reliefs. The input 3D scene is decomposed into two layers encoding the level-of-depth and detailed features respectively. Users can edit each layer separately and combine them to form a seamless bas-relief.
- Our approach allows for changing the styles of bas-relief by editing the depth or normal maps in real time. The feedback is instant, which is quite useful for obtaining a series of bas-reliefs with visually different appearances for a given 3D scene.
- Our system provides a real-time artistic tool for bas-relief modeling. The user only needs to focus on specifying a small number of control parameters.

2. Related Work

We briefly summarize state-of-the-art approaches in this section. Cignoni et al. [CMS97] treated bas-relief generation as a problem of compressing the depth of a 3D scene onto a view

plane. Their principle rule is treating the 3D scene as a height field from the point of view of the camera, which is followed by the subsequent literature and our approach. An advantage of this treatment is that we can easily borrow some approaches developed for tone mapping of High Dynamic Range (HDR) images [FLW02]. For bas-reliefs, depths take place of the intensities in HDR image. Weyrich et al. [WDB*07] proposed an HDR-based approach for constructing digital bas-reliefs from 3D scenes. They did not compress the depths directly, but nonlinearly compress the gradient magnitude to remove depth discontinuities. Kerber et al. [KBS07] proposed a feature preserving approach combined with linear rescaling and unsharp masking of gradient magnitude. An improvement on this approach is proposed in [Ker07], which rescales the gradient nonlinearly, Using four parameters, one can steer the compression ratio and the amount of details to be perceivable in the output. Kerber et al. [KTZ*09] also presented a filtering approach which preserves curvature extrema during the compression process. Song et al. [SBS07] generated bas-reliefs on the discrete differential coordinate domain, combining the concepts of mesh saliency, and shape exaggeration. Bian and Hu [BH11] proposed an approach based on gradient compression and Laplacian sharpening, which produces bas-reliefs with well-preserved details. Inspired by the relations between histogram equalization and bas-relief generation, Sun et al. [SRML09] presented an approach base on adaptive histogram equalization, which provides a new algorithm on bas-relief generation. This approach produces high quality bas-relief and preserves surface features well. Li et al. [LWYM12] presented a two-scale approach for bas-relief estimation from a single image, aiming at restoring brick and stone relief from their rubbing images in a visually plausible manner. Wu et al. [WMR*13] developed an approach of producing bas-reliefs from human face images. They first created a bas-relief image from a human face image, and then used shape-from-shading (SfS) approach on the bas-relief image to construct a corresponding bas-relief. They trained and used a neural network to map human face images to bas-relief images, and applied image relighting technique to generate relit human face images for bas-relief reconstruction. Sýkora et al. presented an interactive approach for generating bas-relief sculptures with global illumination renderings of hand-drawn characters using a set of annotations [SKv*14]. Recently, Ji et al. [JMS14] presented a novel modeling technique for creating bas-reliefs in normal domain. The predominant feature of their approach is that it is able to produce different styles of bas-reliefs and permits the design of bas-reliefs in normal image space rather than in object space.

The previous work from 3D scenes to bas-reliefs can produce high-quality or acceptable results. As explained above, most of them failed to create bas-reliefs in real time. The computational cost of the algorithm which converts the 3D model to the bas-relief is the crucial factor of real-time bas-relief designing. Zhang et al. [ZZZY13] presented a different algorithm based on detail extraction and geometric compression in object space. Their approach reduced the computational cost comparing with traditional work on CPU. Kerber et al. [KTB*10] proposed two algorithms on bas-relief generation in real time. They implemented the full algorithm on graphics hardware and achieved real-time performance.

All the above approaches were built upon the given 3D input scenes. However, only one part of the input 3D scene are exploited, such as the depth, the geometry or the normal. Based on the merits of existing techniques, we develop a bas-relief modeling technique with intuitive stylizing at interactive rates in this paper, making full use of the depth and normal information from the input 3D scenes.

3. Bas-Reliefs from Depth-and-Normal

3.1. Algorithm Overview

The fundamental problems in bas-relief modeling from 3D scenes include preserving the appearance for orthogonal views and squashing the depth gaps [WDB*07]. The previous work [WDB*07, Ker07, KTZ*09, SRML09, BH11, JMS14] has solved them to some extent. However, traditional 3D model-based bas-relief modeling algorithms are often limited to high computational cost or monotonic styles. To improve this work in these directions, we decompose bas-relief generation into two scales:

- extracting a base surface from the depth map using a simple non-linear compression and a box filter;
- restoring fine details from the normal map, with a few intuitive artistically relevant parameters to control the styles.

The involved operators can be implemented in parallel, making our algorithm an ideal candidate for acceleration via GPU implementation. The algorithmic pipeline of our technique is similar to the one of deferred shading technique in the field of 3D computer graphics. We design an OpenGL application to implement the full algorithmic pipeline which is composed of two stages, including decomposition and compression. Figure 1 demonstrates the flowchart of our algorithm, illustrating two stages and how the GPU pipeline is used for real-time bas-reliefs generation. The shading step provides an instant interactive feedback to the user. The user can interactively manipulate the parameters using GUI widgets. The parameters can be dynamically selected, allowing for instant comparison between different styles. This type of interactive feedback is very useful for rapid visual evaluation.

The parallel nature of our algorithm is highly suitable to exploit the properties of GPU. Based on GPU, we developed a high-performance algorithm at interactive rates. The whole framework is conceptually simple, intuitive and easy to implement and use. An example is shown in Figure 2. Due to decomposing the 3D model to two layers, we obtained a bunny bas-relief model with the relative importance of high frequency details enhanced.

3.2. Scene Input and Decomposition

Given a 3D input scene, we do not start from the depth buffer of the scene like the previous work. From 3D models we can obtain the depth information as well as other important geometric information, such as normals. Most of previous work only exploit the

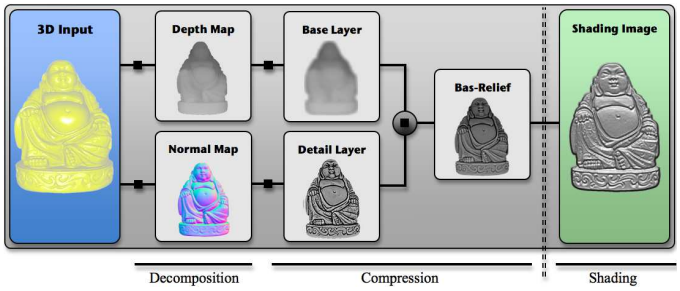
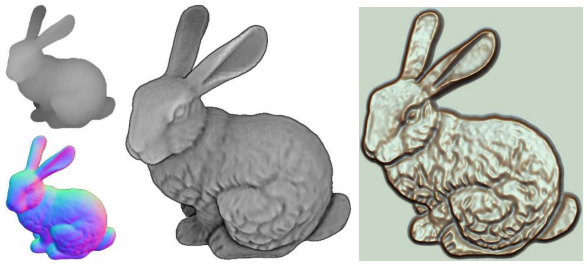


Figure 1: The flowchart of our bas-reliefs generation on the GPU.



(a) (b) (c)

Figure 2: An example generated using our bas-reliefs modeling tool. (a) The original depth and normal maps; (b) the final output texture encoding the bas-relief; (c) the resulting bas-relief surface.

depth map, and the recent work [JMS14] only exploits the normal map. We focus on generating bas-reliefs by combining the depth map with the normal map. At the first stage of our pipeline, we decompose the 3D input scene into two items, a depth map and a normal map with different purposes respectively.

- Such a depth map can be provided by the depth buffer of an OpenGL application by default.
- The normal map is not provided by the OpenGL pipeline by default. We use OpenGL Shading Language to render the normal map in OpenGL context. In this paper, the normal vector is always described in camera space, and the z components of visible normals in camera space are always positive.

We render the involved two maps into G-buffers at one pass on the GPU like the deferred shading technique. As the result at this stage, we obtain two textures containing the depth map and the normal map respectively.

3.3. Base Surface Generation

The goal of bas-relief generation is to compress large magnitude changes in depth, while preserving local changes of small magnitude as much as possible. In general, the depth map exhibits high jumps in boundary regions and its occlusion areas, thus special methods are introduced to remove these large gaps in previous work. Most previous gradient-based methods successfully removed the depth gaps, and eliminated the level-of-depth as well.

(a) (b) (c)

Figure 3: Base surface obtained through the compression along with a low-pass filter. (a) The original depth map; (b) the corresponding mesh surface of (a); (c) the resulting base surface.

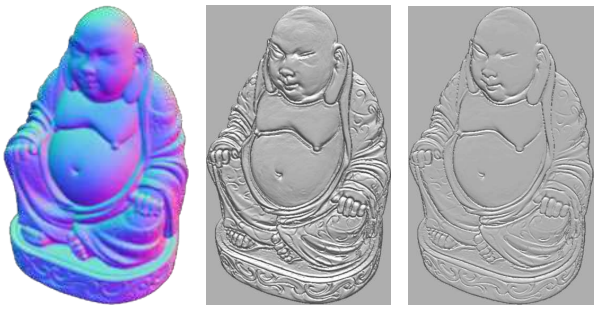
We use a simple non-linear compression along with a low-pass filter to fulfill this task. With small magnitude changes and smooth appearance, the compressed depth encodes the global and coarse structure of the underlying relief which is regarded as a **base surface** $\mathcal{B}(u,v)$ in our approach. In our test, a global compression via a nonlinear compression function to the depth works well, because it is not necessary to preserve local features for generating the base surface. The range of the depth map $\mathcal{H}(u,v)$ is normalized into $[0,1]$ by default in OpenGL, so we rescale the depth map as follows,

$$\mathcal{H}'(u,v) = \mathcal{H}(u,v)/(1+\beta\mathcal{H}(u,v)), \qquad (1)$$

where $\beta >= 0$. After rescaling, a low-pass filter is used for further smoothing the jumps of the rescaled depth map $\mathcal{H}'(u,v)$. To eliminate the depth jumps isotropically, we use the box filter which is a simple average of all the pixels inside a square region. Users can specify the filter kernels as $(2n+1)\times(2n+1)$ rectangular arrays of pixel contributions, and all values of the filter kernel are assigned the same weights $1/(2n+1)^2$. Users can choose the kernel size n according to the image resolution, and pass it directly as a parameter to the shader. The box filter is implemented as a two-pass convolution, which does not sacrifice the efficiency even with a big size (like $n = 20$) in our experiments. An example is given in Figure 3. Figure 3(a) and Figure 3(b) show the original depth map, and Figure 3(c) shows a smooth base surface obtained through a nonlinear scaling along with a box filter. In this example, we set $\beta = 2$ and $n = 3$. Our simple approach eliminates the high jumps at boundary and occlusion regions effectively. However, this filter meanwhile diminishes local details which will be extracted from normal map and added into the base surface to generate a bas-relief as described in the following section.

3.4. Detail Peeling

Control over detail is a central requirement when designing a bas-relief. Accordingly, to preserve local details for orthogonal views of the given 3D input scenes is a key step in the compression stage



(a) (b) (c)

Figure 4: An example for Scheme I using different scaling functions. (a) A normal map; (b) function $\mathscr{F}_1(x) = x$ nearly preserves the original appearance; (c) and the result from function $\mathscr{F}_1(x) = \sqrt{x}$ conveys a flattened impression.

of our pipeline. We address this by introducing two kinds of detail peeling operators in the following subsections.

3.4.1. Scheme I

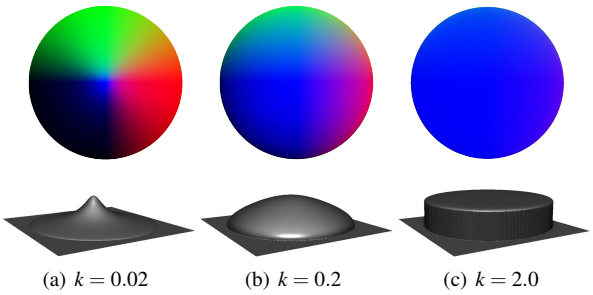
It is well known that normals play an important and essential role in real-time rendering. Normal is also essential in the inverse problems of rendering, such as photometric stereo, shape from shading [Woo84] and 3D modeling [WTBS07]. Our work is somewhat related to these work, but our goal is to create bas-reliefs under a height constraint and free from depth gaps. The normal can directly play a role as the feature detail in our bas-reliefs generating pipeline. Given a normal vector $\boldsymbol{N}^c = (\boldsymbol{N}^c_x, \boldsymbol{N}^c_y, \boldsymbol{N}^c_z)$ in camera space, the z components of visible normals $\boldsymbol{N}^c_z$ are always positive. Utilizing the normal straightforwardly, we define the feature detail for orthogonal views peeled off the 3D shape as follows,

$$\mathscr{D}_1(u,v) = \mathscr{F}_1(\boldsymbol{N}^c_z(u,v)) \tag{2}$$

where $\boldsymbol{N}^c_z(u,v)$ is the the z components of the visible normal, $\mathscr{F}_1(x)$ is a scaling function which can be a nonlinear compression to the scalar magnitude, largely boosting small details while attenuating large ones. Figure 4(b) and 4(c) show an example for different scaling functions applied to the same input scene Figure 4(a). For this example, we used scaling functions $\mathscr{F}_1(x) = x$ (figure 4(b)) and $\mathscr{F}_1(x) = \sqrt{x}$ (Figure 4(c)) to generate different effects.

Re-normaling. The above presented Equation 2 already provides an effective automated detail generation from 3D scenes. However, it only accesses the z components of normals without taking the x and y components into account.

Furthermore, we manage to make our approach allow for additional intuitive editing operations depending on artistic demands. A fundamental insight in bas-relief is that the changing of normal strongly influences the appearance of a bas-relief. For instance, raising the z component N^c_z and renormalizing the new normal vector generates a flatten effect. A family of functions $\mathscr{F}_1$ in Equation 2 can be used to modify the normals directly while holding the geometry of the 3D scene. Alternatively, we scheme out a simple formula to control over the surface features via tweaking a couple of parameters. First, we forwardly modify the z component of the normal vector as follows,



(a) $k = 0.02$ (b) $k = 0.2$ (c) $k = 2.0$

Figure 5: Users can intuitively control over the appearance styles of the resulting bas-relief by tweaking one parameter in real time.

$$\bar{N}^c_z(u,v) = \frac{\alpha k N^c_z(u,v)}{1 + k N^c_z(u,v)} \tag{3}$$

where $k > 0$ is used for the non-linear compression, $\alpha > 0$ is for scaling the result ($\alpha = 1$ by default), the combination of them are for controlling over the sharp or flatten effects. Then we update the normal by normalizing the vector $(N^c_x, N^c_y, \bar{N}^c_z)$. Our re-normaling scheme applies a nonlinear compression function to the z component N^c_z indirectly, inducing to attenuate or boost the surface slopes. We found that it provides reasonable effects as seen from the example shown in Figure 5. Given an input sphere model, users can easily turn out a cusp-like or a disk-like result by tweaking a proper value of the parameter k. In Figure 5, the upper row shows the updated normal maps and the lower row shows the resulting height fields corresponding to the z component of the updated normals.

To follow and improve the stylizing process proposed in [JMS14], we address this by including three intuitive styles which exhibits sharp, round and flat impressions respectively. The stylization problem in previous work is accomplished by formalizing the problem as a quadratic optimization, which results in solving a sparse linear system. Therefore, the resulting style can not be visible quite instantly. In our pipeline, we present an interactive scheme to produce different styles in real time. The stylization effects of changing parameters are feeding back without delay. In this way, the style generation of bas-relief becomes interactive. Figure 6 gives an example, showing that our approach allows for constructing relief surfaces with different visual appearances by controlling a couple of parameters.

It is worth noting that we only exploit the normal information here, without compressing the depth map. Our simple trick provides reasonable results without eliminating important features as seen from the figures. Generally speaking, normals from a single view cannot be utilized to restore the underlying fully extended

(a) $k = 0.3, \alpha = 1.0$ (b) $k = 1.0, \alpha = 1.0$ (c) $k = 1.0, \alpha = 10.0$

Figure 6: Bas-relief generation with different visual appearances though selecting a couple of parameters. The most advance of our scheme is that the feedback is instant in our GPU-based pipeline.

3D objects directly. However, details peeling off normals is adequate to the bas-relief generating which focuses on the feature details for orthogonal views.

Remarks. As can be seen from the above examples, by turning the z-components of the visible normal, users can easily obtain multiple detail styles, which is our motivation to introduce this scheme. In this paper, we focus on approximating or simulating local details from normals, rather than reconstructing the geometry globally. We roughly treat the variation of the normal z-component as the detail under an orthographic view. Generally speaking, it may also produce extra details (see the example shown in Figure 11) into the resultant bas-relief. Figure 4 and Figure 11 show that this scheme may generate results with the effect of carving along the occlusion edges.

This scheme has its limitation. For example, if the input shape is a cone with the axis in z-direction, then all normal z-components are constant, and the produced details are also constant. However, the cone shape has been retained in the base surface generated above. This scheme is still useful for two reasons. Firstly, although using normal directly as height values may not faithfully reproduce the original shape surface, it can still produce similar diffuse shading to the original surface in some cases, if the light direction is parallel to z axis. Secondly, the extracted details using this scheme are used to simulate the visual effects in a visually plausible manner, not to preserve its shape features faithfully. In fact, the global shape of the bas-relief is determined by the base surface discussed in Section 3.3, and the combination of the base surface and the details will be discussed in Section 3.5.

As an alternative, another scheme to extract more faithful feature details is presented in the following subsection.

3.4.2. Scheme II

Besides the pixelwise scheme described above, we introduce another scheme to define local detail by taking neighboring pixels into account. Given a normal map $\boldsymbol{N}^c$ obtained at the decomposition stage, we compute the details similarly to [JMS14] as follows,

$$\mathcal{D}_2(u,v) = g(Div((\boldsymbol{N}^c_x/\boldsymbol{N}^c_z, \boldsymbol{N}^c_y/\boldsymbol{N}^c_z)), \boldsymbol{N}^c), \tag{4}$$

where function g is used to screen out the fine details from the given normal map. g can be a truncated function involving a threshold to trim away the outliers. To attenuate the spurious features especially along the the steep boundaries and occlusion areas, we define g as follows,

$$g(x, \boldsymbol{N}^c) = x \cdot \boldsymbol{N}^c_z \tag{5}$$

where $\boldsymbol{N}^c_z$ is small at the occlusion areas. On the other hand, the boundaries and occlusion areas are already preserved on the base surface to a certain extent. This simple function worked well for bas-reliefs generation in our experiments. Then we use backward difference to define a discrete analog of the divergence operator,

$$\begin{aligned} Div(X) = {} & X_u(u,v) - X_u(u-1,v) \\ & + X_v(u,v) - X_v(u,v-1). \end{aligned} \tag{6}$$

It is worth while to note that we compute the detailed features using the normal map rather than the depth map. The discontinuities in the depth map of the input scene will introduce large gradient discontinuities, so the users have to remove the discontinuities by setting an interval value explicitly. Another advantage of our normal-based method is that allows editing the details intuitively (like the re-normaling operator in Scheme I).

3.5. Post-processing

At this stage we have obtained two layers, a base layer and a detailed layer which are both in the form of textures on the GPU. The base layer encodes the coarse surface extracted from the depth map, and the detailed layer encodes the local details extracted from the normal map. We need to combine both layers to retain the final bas-relief with a global structure and visually important features. This results in a new height field with adjusted weights for the different layers. This step is similar to the one of previous work like [LWYM12] which aimed at restoring brick and stone reliefs from their rubbing images. However, the motivation of our method is to present a real-time solution to the problem from 3D scenes to bas-reliefs. To this end, our approach involves only local computations for both the base surface and the detailed features, which is different from the previous work involving a large linear or non-linear system. Finally, we combine these two layers in an additive way as follows,

$$\mathcal{R}(u,v) = \mathcal{B}(u,v) + \lambda \mathcal{F}_2(\mathcal{D}(u,v)). \tag{7}$$

The above equation contains only one user defined parameter λ to balance these two items. Now that the detailed layer contains the high-frequency small-scale features of the surface, we will further boost or attenuate their influence relative to the coarse shape in the base layer by introducing a function which is set $\mathcal{F}_2(x) = x$ by default. To suppress the sharp edges (such as the boundaries and occlusions) and to boost the weak features, we introduce a non-linear blending $\mathcal{F}_2(x) = x/(\varepsilon + \sqrt{|x|})$. However, the linear or non-linear combination of two layers may introduce

(a) (b) (c)

Figure 7: Bas-relief obtained by combining two layers. (a) The original depth map which exhibits high jumps at boundary and occlusions regions; (b) base surface extracted from (a); (c) add local details into the base surface to form the final bas-relief.

artifacts in the resulting bas-relief. To eliminate the artifacts, we use a Gaussian blur filter to extract the final bas-relief. An example is given in Figure 7. We set the parameters $\lambda = 1$ and $\mathscr{F}_2(x) = x$ in this example. The high jumps at occlusion regions in the original depth map were eliminated as well as the visually important details (see Figure 7(b)). Scaling each layer and recombining them by above parameter to yields a modified height field with important features preserved (see Figure 7(c)).

4. Implementation and Results

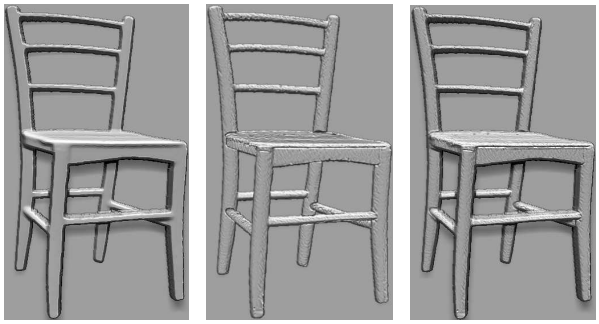
4.1. Implementation

As explained above, we designed the converting procedure exploiting the parallel character of the modern GPUs. To efficiently exploit the properties of GPU, we convert the 3D data to the height and the normal in a local way. The pipeline of our algorithm can be fully implemented in the OpenGL context on modern GPUs. This results in a single application that achieves real-time performance and allows the user to seamlessly control the meaningful parameters as well as transform objects in the scene. The effect of changing parameters is visible instantly, thus the whole process of bas-relief generation is truly at interactive rate.

In our implementation, we render all intermediate results into float-point textures with a resolution of 800x800 pixels, and we render them into separate textures at once using a capability of OpenGL called Multiple Render Targets (MRT). The performance is nearly independent of the complexity of our tested scenes (with 7k to 870K vertices), and it achieves frame rates about 28.9 FPS averagely. The performance was measured using an NVidia GeForce 9800 GT and a 3.00GHz Intel Core 2 Duo CPU E8400 with 2GB RAM, in OpenGL Shading Language.

4.2. Results

As mentioned above, one of our goals is to reconstruct a height field under a height constraint so that the appearance looked in front of the bas-relief should appear similar to that of the input scene. We have experimented our approach on a variety of 3D scenes. In all cases, our approach is capable of producing perceptually plausible digital bas-reliefs without much parameter tweaking. The following figures are given to demonstrate these results.



(a) (b) (c)

Figure 8: Bas-relief obtained by combining depth-and-normal maps. (a) The base surface from the depth map only; (b) the detailed layer extracted using the normal map only; (c) add local details into base surface to form the final bas-relief ($\lambda = 1$).

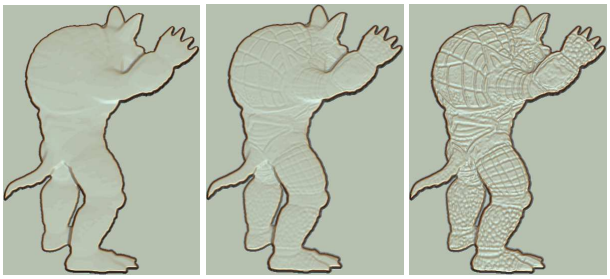
(a) (b) (c) (d)

Figure 9: Bas-reliefs generated by our approach with different compression over the depth. (a) The resulting height map using parameters $\beta = 0$, $\lambda = 0.3$; (b) the bas-relief from the height map (a); (c) the resulting height map using parameters $\beta = 4$, $\lambda = 0.3$; (d) the bas-relief from the height map (c).

The example in Figure 8 shows that the effects of decomposition and combination of two layers. The left image was produced using only the depth map, the middle image was produced using the normal map, and the right image was resulted from the combination of two maps. The detailed layer (shown in Figure 8(b)) was extracted using the Scheme I without re-normaling. Although the detailed layer can be regarded as a flattened bas-relief, the combination result displays the level-of-depth.

The example in Figure 9 shows that the effects resulting from different compressions over the depth map. As can be seen in the figures, a smaller value of β accentuates the original depth map and maintains the perceptually salient parts, while a bigger value of β induces a higher compression over the bigger depth and results in a smaller range of depths.

By tweaking the parameters λ in Equation 7, user can create a series of bas-reliefs with different levels of detail. An example is shown in Figure 10. The detailed layer (used in Figure 10(b) and Figure 10(c)) was extracted using the Scheme II. Due to the separate extraction of feature details and base surface, the visual



(a) (b) (c)

Figure 10: Bas-relief obtained by combining two layers. (a) The base surface; (b) adding details into the base surface to form a bas-relief ($\lambda = 0.1$); (c) adding details into the base surface to form a bas-relief ($\lambda = 0.5$).

(a) (b) (c) (d)

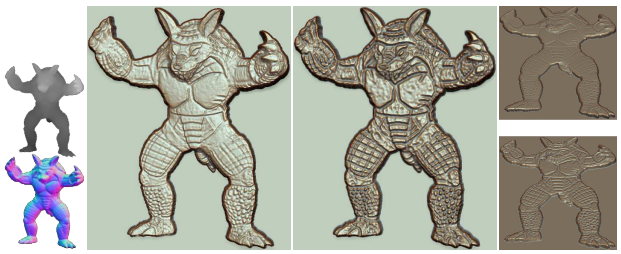
Figure 11: Comparisons between two detail schemes testing on a noisy model. (a) Scheme I ($\lambda = 0.2$); (b) Scheme II ($\lambda = 0.2$); (c) bas-reliefs generated from a different view of the same model; (d) a tilted view of (a) and (b).

cues remain perceivable in resulting bas-reliefs even with a small range of height. Our approach provides certain freedom to design bas-reliefs with different levels of detail at interactive rates.

As described above, we scheme out two formulas to encode the local details. Figure 11 illustrates comparisons between these two schemes. There are slight differences between them as seen from these images. In comparison with Scheme I, Scheme II keeps a sharp and faithful impression of small details (also faithfully preserved the heavy noises throughout the original 3D model) for the same parameters setting since it evaluates the detail by accessing the neighboring pixels. Scheme I converted the z component of the normal to the depth value, which may introduce some visible artifacts or called unfaithful details, such as the rings on the right wing and some occlusion edges. The causation of this effects is that the z components are insensitive to the ridges and valleys. It might not be accepted to reconstruct faithfully objects in 3D space. However, as seen from the figures, Scheme I generated visibly reasonable bas-reliefs impressing ones with the effect of carving along the occlusion edges (see Figure 11(a)). Users may choose the schemes and further edit the details by altering the parameters to meet their artistic demands.

(a) (b) (c) (d)

Figure 12: Examples for Scheme I using different values of the parameter k. (a) The depth and normal maps of the input 3D scene; (b) the resulting bas-relief using the parameter setting $k = 2$ in Equation 3; (c) the resulting bas-relief using the parameter setting $k = 0.2$ in Equation 3; (d) a tilted view.



(a) (b) (c) (d)

Figure 13: Examples for layers blending via different functions $\mathscr{F}_2(x)$ ($\lambda = 0.2$). (a) The depth and normal maps of the input 3D scene; (b) the resulting bas-relief using the function $\mathscr{F}_2(x) = x$ in Equation 7; (c) the resulting bas-relief using the function $\mathscr{F}_2(x) = x/(\varepsilon + \sqrt{|x|}), \varepsilon = 10^{-6}$ in Equation 7; (d) a tilted view.

The following figures displayed various appearances through different parameters and functions. For the example shown in Figure 12, we controlled over the detail using the Scheme I. A larger value of k in Equation 3 turns out more flattened appearance. In Figure 13, we showed an example using different functions $\mathscr{F}_2(x)$ in Equation 7. The non-linear function $\mathscr{F}_2(x) = x/(\varepsilon + \sqrt{|x|})$ enhanced the detailed features exaggeratedly. Our algorithm provides certain freedom to design bas-reliefs with visually various styles at interactive rates.

More examples are shown in Figure 14. We can see the bas-reliefs shown in this figure preserve very fine details with a smaller range of depths compared to the original scene, without losing the level-of-depth (see Figure 14(a)-14(c)).

Parameter exploration. We briefly list the intuition behind the parameter setting hereinafter. The parameter β in Equation 1 is used to control the level-of-depth. A larger value of β removes the gaps of the depth map more effectively, but loses more of the level-of-depth and produces a more flattened result; a small value approximates a linear scaling of the depth and generates a result with more levels of depth. We set $\beta = 0.3$ by default. By

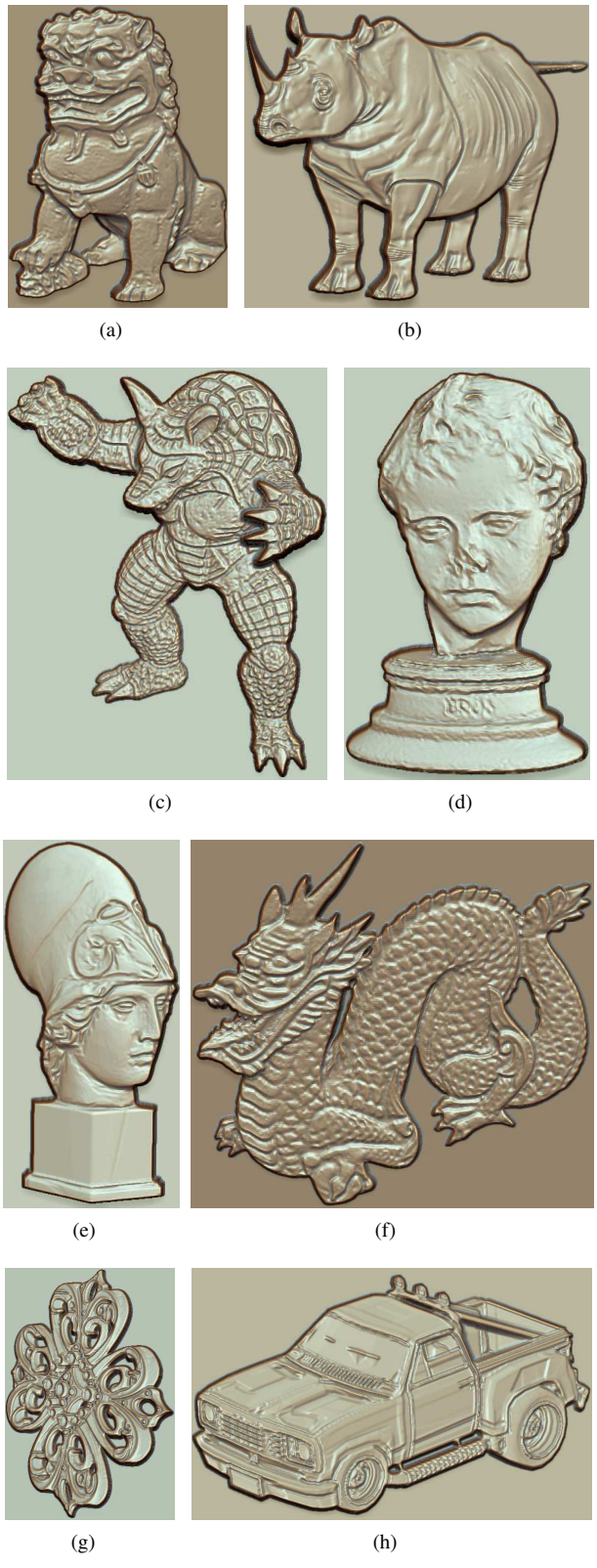


Figure 14: More bas-reliefs generated using our approach.

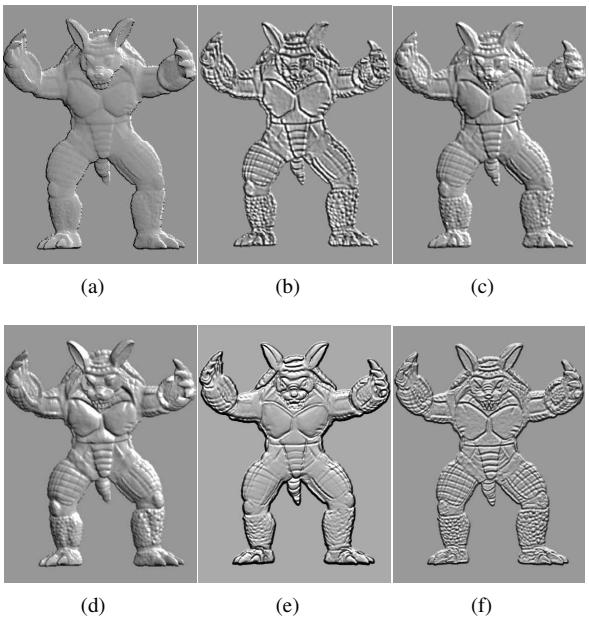


Figure 15: Bas-reliefs produced by the approaches of Cignoni et al. [CMS97] (a), Kerber et al. [KBS07] (b), Kerber [Ker07] (c), Sun et al. [SRML09] (d) and our approach using Scheme I ($\lambda = 1$) (e), our approach using Scheme II ($\lambda = 1$) (f).

tuning the parameter λ in Equation 7, the user can intuitively control how much the fine details appear in the resulting bas-relief. Higher values of λ imply that more details will remain perceivable in the bas-relief. We set $\lambda = 0.2$ by default. The re-normaling formula (Equation 3), and augment functions $\mathscr{F}_1$, $\mathscr{F}_2$ are optional. To sharpen or flatten the details when using the first scheme, the user can tweak the parameter k in Equation 3. In our experiments, k is usually set to values between 0.01 and 2, higher values corresponding to a stronger flattened effect while lower values corresponding to a stronger sharpened effect. If a large λ is inadequate to emphasize the detailed features, the user may further set these functions as above mentioned to enhance the detailed effect. Table 1 lists a table about the complexity, frame rates and important parameters of most examples shown in the paper. As we can see, the performance of our approach is nearly independent of the complexity of our tested models.

4.3. Comparison

Now we turn to compare our approach with other CPU-based bas-relief generation approaches. We begin by comparing our results with those of Cignoni et al. [CMS97], Kerber et al. [KBS07, Ker07], and Sun et al. [SRML09]. Results for both of Kerber's approches were obtained using default parameter settings ($\tau = 5$, $\sigma_1 = 4$, $\sigma_2 = 1$, $\alpha = 4$, and the compression ratio is 0.02). Results for Sun's approach were obtained using the following parameters: $B = 10000$, $m_0 = 32$, $n = 4$, $l = 16$, and $K = 1$. Figure 15 shows results using the above mentioned approaches.

Model	Vertices	FPS	Important Parameters
Figure 8	212634	28.9	$\beta = 0, \lambda = 1$
Figure 10(c)	172974	28.9	$\beta = 0.5, \lambda = 0.5$
Figure 11(a)	863210	28.9	$\beta = 0.5, \lambda = 0.2$
Figure 12(c)	195586	28.9	$\beta = 0.3, \lambda = 0.25$
Figure 13(c)	172974	28.9	$\beta = 0.3, \lambda = 0.2$
Figure 14(d)	55375	28.9	$\beta = 0.3, \lambda = 0.2$
Figure 14(e)	7546	28.9	$\beta = 0.3, \lambda = 0.2$
Figure 14(f)	437645	28.9	$\beta = 0.3, \lambda = 0.2$

Table 1: A table showing the complexity, frame rates and important parameters for some typical examples shown above. We set $\mathscr{F}_2(x) = x/(\varepsilon + \sqrt{|x|}), \varepsilon = 10^{-6}$ for the examples shown in Figure 13(c) and Figure 14(d)-14(f); and set $k = 0.2$ for Figure 12(c).

Except Cignoni's approach, other approaches produced natural or feature enhanced results. Our approach provides different types of results with natural or exaggerated features. Due to its whole pipeline remains on GPU, our approach achieves real-time performance. An execution of the entire pipeline of our approach including the shading step elapsed about 18 milliseconds, while other CPU-base approaches expended about several seconds in our experiments.

5. Conclusion and Future Work

We presented a real-time interactive tool designed to support artists and enthusiasts in creating bas-relief using 3D scenes. Beside the capability of producing bas-reliefs in real time, we demonstrated that our approach is also suitable to create bas-reliefs with different styles.

The main contributions of the paper consist of enlarging the palette of bas-relief generation schemes and using a GPU-based acceleration scheme. Unlike most of the previous work, our approach generates bas-reliefs from 3D scenes without solving a large sparse system of linear equations. And the parallel nature of our algorithm is highly suitable to exploit the properties of GPU.

Our work focused on creating bas-reliefs using depth-and-normal maps currently. However, it would be easy and useful to add more properties into our real-time pipeline, such as lines, textures, etc. The lines can be extracted in object space or in image space, and the textures will be attached to create colorful bas-relief which reflects different materials. Such work will be investigated in our future work.

Acknowledgement

We would like to thank the anonymous reviewers for their constructive comments. This work was partially supported by the National Natural Science Foundation of China (61202278), the Zhejiang Provincial Natural Science Foundation of China (Y1111101), the Defense Industrial Technology Development Program of China and EPSRC (EP/J02211X/1).

References

[BH11] BIAN Z., HU S.-M.: Preserving detailed features in digital bas-relief making. *Computer Aided Geometric Design 28*, 4 (2011), 245–256. 2

[CMS97] CIGNONI P., MONTANI C., SCOPIGNO R.: Computer-assisted generation of bas- and high-reliefs. *J. Graph. Tools 2*, 3 (1997), 15–28. 1, 8, 9

[FLW02] FATTAL R., LISCHINSKI D., WERMAN M.: Gradient domain high dynamic range compression. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 249–256. 2

[JMS14] JI Z., MA W., SUN X.: Bas-relief modeling from normal images with intuitive styles. *IEEE Transactions on Visualization and Computer Graphics 20*, 5 (2014), 675–685. 2, 4, 5

[KBS07] KERBER J., BELYAEV A., SEIDEL H.-P.: Feature preserving depth compression of range images. In *Proceedings of the 23rd spring conference on computer graphics* (2007), Budmerice, Slovakia, pp. 110–114. 2, 8, 9

[Ker07] KERBER J.: *Digital Art of Bas-Relief Sculpting*. Masters thesis, Universität des Saarlandes, August 2007. 2, 8, 9

[KTB*10] KERBER J., TEVS A., BELYAEV A., ZAYER R., SEIDEL H.-P.: Real-time generation of digital bas-reliefs. *Journal of Computer-Aided Design and Applications 7*, 4 (2010), 465–478. 2

[KTZ*09] KERBER J., TEVS A., ZAYER R., BELYAEV A., SEIDEL H.-P.: Feature sensitive bas relief generation. In *IEEE International Conference on Shape Modeling and Applications Proceedings* (Beijing, China, June 2009), IEEE Computer Society Press, pp. 148–154. 2

[LWYM12] LI Z., WANG S., YU J., MA K.-L.: Restoration of brick and stone relief from single rubbing images. *IEEE Transactions on Visualization and Computer Graphics 18*, 2 (2012), 177–187. 2, 5

[SBS07] SONG W., BELYAEV A., SEIDEL H.-P.: Automatic generation of bas-reliefs from 3d shapes. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications* (2007), pp. 211–214. 2

[SKv*14] SÝKORA D., KAVAN L., ČADÍK M., JAMRIŠKA O., JACOBSON A., WHITED B., SIMMONS M., SORKINE-HORNUNG O.: Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transaction on Graphics 33*, 2 (2014), 16. 2

[SRML09] SUN X., ROSIN P. L., MARTIN R. R., LANGBEIN F. C.: Bas-relief generation using adaptive histogram equalization. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (2009), 642–653. 2, 8, 9

[WDB*07] WEYRICH T., DENG J., BARNES C., RUSINKIEWICZ S., FINKELSTEIN A.: Digital bas-relief from 3d scenes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), p. 32. 2

[WMR*13] WU J., MARTIN R. R., ROSIN P. L., SUN X., LANGBEIN F. C., LAI Y.-K., MARSHALL A. D., LIU Y.-H.: Making bas-reliefs from photographs of human faces. *Computer-Aided Design 45*, 3 (2013), 671–682. 2

[Woo84] WOODHAM R. J.: *Photometric Method for Determining Shape from Shading*. Tech. rep., 1984. 4

[WTBS07] WU T.-P., TANG C.-K., BROWN M. S., SHUM H.-Y.: Shapepalettes: interactive normal transfer via sketching. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), p. 44. 4

[ZZZY13] ZHANG Y., ZHOU Y., ZHAO X., YU G.: Real-time bas-relief generation from a 3d mesh. *Graphical Models 75*, 1 (2013), 2–9. 2