Appearance-Mimicking Surfaces

Christian Schüller Daniele Panozzo Olga Sorkine-Hornung

ETH Zurich



Figure 1: *A collection of appearance-mimicking surfaces generated with our algorithm.*

Abstract

We consider the problem of reproducing the look and the details of a 3D object on a surface that is confined to a given volume. Classic examples of such "appearance-mimicking" surfaces are bas-reliefs: decorations and artwork depicting recognizable 3D scenes using only a thin volumetric space. The design of bas-reliefs has fascinated humankind for millennia and it is extensively used on coins, medals, pottery and other art forms. We propose a unified framework to create surfaces that depict certain shapes from prescribed viewpoints, as a generalization of bas-reliefs. Given target shapes, viewpoints and space restrictions, our method finds a globally optimal surface that delivers the desired appearance when observed from the designated viewpoints, while guaranteeing exact, per-vertex depth bounds. We use 3D printing to validate our approach and demonstrate our results in a variety of applications, ranging from standard bas-reliefs to optical illusions and carving of complex geometries.

CR Categories: I.3.5 [Computer Graphics]: Computational geometry and object modeling—Curve, surface, solid, and object repres.

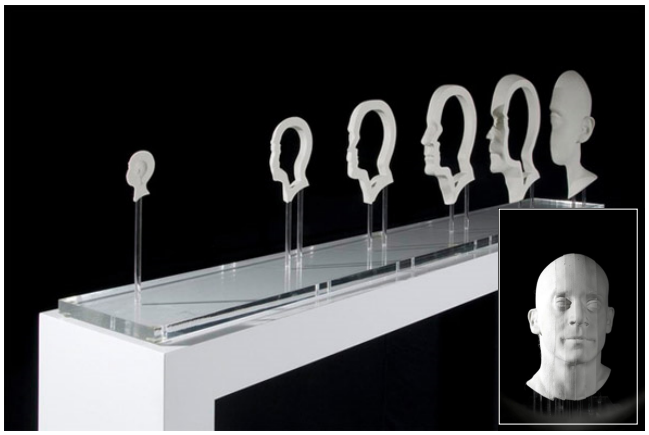
Keywords: bas-relief, convex optimization, spatial constraints

Links: DL PDF WEB VIDEO

1 Introduction

Bas-reliefs are thin surfaces whose normals resemble the normals of a different surface or a general 3D scene, giving a (false) impression of depth when observed from the right viewpoint. Bas-reliefs have been used for centuries in artistic masterpieces, and are ubiquitous on coins and medals. The most common bas-reliefs are thin layers of stone or ceramic covering a single object, but they can also be fragmented into disconnected slices to obfuscate the shape, creating interesting optical illusions (Figure 2).

The design of bas-reliefs has been a subject of interest in computer graphics in the past two decades. A bas-relief is essentially a 2.5D



(c) Jonty Hurwitz, 2014

Figure 2: *We draw inspiration from the art of Jonty Hurwitz, building sculptures of fragmented and disconnected slices to obfuscate the shape and creating interesting optical illusions.*



Figure 3: *Inspired by street artwork painted over the steps of a staircase, we use our algorithm to embed a 3D model of an owl into a staircase. Constrained to a thin layer, the relief does not affect the function of the staircase, while being much more resistant than paint to erosion and aging.*

image, which has a strong relation with the depth buffer used in the standard graphics pipeline. Most works have proposed to create bas-reliefs from given digital 3D scenes by either directly compressing the depth buffer of the scene's rendering or by working in the gradient domain, where the final model is obtained by solving a Poisson equation.

In this work, we define *appearance-mimicking surfaces* (AMS) that generalize bas-reliefs, lifting their restriction to a height field. Our generalization makes the reliefs usable at a wider range of viewing angles, while still guaranteeing self-intersection free results, which is mandatory for subsequent fabrication.

Specifically, we develop a mathematical framework to compute surfaces whose normals optimally approximate the normals of a given 3D shape or scene, while strictly obeying given depth- or volume-confinement constraints. Direct fitting of normals and spatial constraints is in general a challenging, nonlinear problem, which led previous works to employ heuristics that circumvent difficult numerical optimizations. Unfortunately, giving up the constrained optimization of normals means forfeiting bounds on geometry and appearance distortion in the resulting relief. Instead, we propose a novel view-dependent surface representation which allows us to cast the optimization as a quadratic program. The resulting problem formulation is convex, and we are guaranteed to find the optimal solution under feasible constraints.

Differently from previous works, our method does not rely on rasterization of the input geometry and the depth buffer. AMS are generated by deforming the input mesh without modifying its connectivity, thereby increasing the algorithm's efficiency, details preservation and allowing to easily transfer surface attributes. As a positive side effect of our representation, we can exactly satisfy per-vertex depth constraints and we can "project" the target shapes on disconnected and arbitrarily shaped surfaces, as shown in Figure 3.

Our algorithm is controllable and robust, enabling to design complicated appearance-mimicking surfaces with minimal user effort. We test our method in a variety of applications, such as the design of optical illusions in architectural settings and the creation of carving patterns on complex geometries. To verify the realism of our model and lighting assumptions, we validate our results via 3D printing.

The contributions of this paper can be summarized as follows:

1. We cast the generation of appearance-mimicking surfaces as a quadratic program. Our algorithm is guaranteed to find the unique solution that approximates the shading of the original model.
2. Our method allows versatile depth control, enabling to carve bas-reliefs on complex objects and with precise spatial constraints.
3. We extend bas-reliefs beyond height fields and validate our results via 3D printing.

2 Related work

Height field compression. The digital generation of bas-reliefs was pioneered by Cignoni et al. [1997], who created bas-relief models of given 3D objects by linearly compressing (squeezing) the depth map of their rendering, obtained using OpenGL-based rasterization. By swapping the linear compression with more advanced nonlinear and adaptive scaling functions, it is possible to increase the visual quality [Sun et al. 2009]. Nonetheless, there is no direct connection between the compression of the geometry and the lighting equation: the surface normals may significantly change after the squeezing operation, and the resulting bas-reliefs are prone to looking different from the desired appearance, requiring a heuristic post-processing step to add details and increase the depth illusion.

Gradient field compression and Poisson reconstruction. A breakthrough in the generation of digital bas-reliefs has been proposed in [Weyrich et al. 2007], where instead of compressing the height field directly, modifications are applied to its surface gradients, and a new height field matching the manipulated gradients in the least-squares sense is extracted by solving the Poisson equation (a linear system). Many variants of this algorithm have been explored [Song et al. 2007; Kerber et al. 2009; Bian and Hu 2011; Zhang et al. 2013], with different kinds of filters applied to the gradients, or to the final surface in post-processing. However, all these methods suffer from the intrinsic limitation that the modified gradients are in general not integrable, and the normals of the surface generated in the Poisson step can be far from the desired normals.

Laplacian compression. Ji et al. [2014] propose to directly minimize the L_2 difference between the Laplacian of the bas-relief height field and the depth discontinuity free surface computed from a rendered normal image. This approach allows for artistic editing in the 2D domain and produces higher-quality results than previous methods. However, their formulation is limited to height fields and does not provide exact pointwise control over the depth of the generated bas-reliefs. They use a penalty based approach to control the thickness of the height field which depends on multiple parameters. Furthermore, they only consider an orthographic projection in their optimization. Our approach lifts both limitations, providing realistic results under perspective viewing, as well as precise and fine-grained depth/volume control.

Encoding the height field and depth control. The majority of the methods mentioned above encode the height field as an image, and only few methods work directly on the input 3D geometry representation. While the former approach greatly simplifies the implementation of the algorithm, the latter approach allows to preserve the sharp details in the scene and the original modeling resolution. In this paper, we decided to use a mesh-based approach, but it is straightforward to adapt our algorithm to work on depth images.

To the best of our knowledge, none of the existing methods allow fine-grained control of the depth: They provide a parameter that controls the maximum depth of the bas-relief, but they cannot be



Figure 4: *Appearance-mimicking surface of the* Dragon Head *model constrained to carve a V-shaped, thin geometry.*

used to create bas-reliefs that fill a complex volume shape. Our method can guarantee that the sculpture will be contained within a specific depth volume, specified as a per-vertex range, and it optimally uses the entire available space (Figure 4).

Bas-relief ambiguity. If a surface with Lambertian reflectance is viewed orthographically from a fixed view point, there is a set of transformations of the object's geometry and the corresponding light sources which do not change the perceived shading and self-shadowing of the object, making it impossible for an observer to determine its true geometry. This is known as the "Generalized Bas-relief Ambiguity" (GBA) [Belhumeur et al. 1999] and even holds for slight changes of the viewpoint. Unfortunately, this is not applicable for real world scenarios where the perspective has to be taken into account and in general no assumptions can be made about the position and direction of the illumination. Chandraker et al. [2005] and Tan et al. [2011] analyze (inter-)reflections and show how they can be utilized to overcome the GBA to recover the geometry from photometric stereo where the light source directions and strengths are unknown.

Camouflaging and artistic applications. Embedding multiple objects into a single sculpture has interested many researchers and artists. The 2D version of this problem is known as image camouflaging [Chu et al. 2010], where multiple images are concealed inside a large and complex scene. The first extension to 3D was proposed in [Sela and Elber 2007], where the silhouettes of multiple objects are embedded in a single model that matches some prescribed silhouettes from a set of predefined viewpoints. A similar idea has been used in [Mitra and Pauly 2009] to create shadow-art sculptures. In [Alexa and Matusik 2010], [Bermano et al. 2012] and [Baran et al. 2012], a special surface with an optimized micro-structure is used to display multiple images, which can be selected by changing the lighting conditions. A similar effect is obtained with a completely different technique in [Elber 2010], where multiple images are sliced and encoded in a set of thin pillars which align to form the desired image only in a prescribed viewpoint. Our technique can be used to design bas-reliefs from given 3D models and project them onto complex geometric scenes to camouflage 3D objects (Section 4).

Several works considered inverse problems related to reliefs. Zatzarinni et al. [2009] extract the relief layer from scanned artifacts using robust height function fitting for archeological analysis purposes. Kolomenkin et al. [2011] reconstruct a fitting bas-relief surface of certain thickness given completely flat input in form of line drawings. They employ Laplacian-based surface inflation, where the Laplacian vectors are hallucinated from the given curves. Since in our setting the target shape is given, the exact normals and Laplacians are readily available for fitting.

3 Method

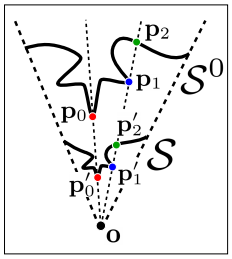
An appearance-mimicking surface is a surface that *looks* similar to another from a fixed perspective, while having a different geometry.

Lighting model. Assuming a Lambertian material with directional lights and no specular component, we can model the color of a surface point $\mathbf{p}$ as:

$$I_{\mathbf{p}} = k_a i_a + \sum_{\mathbf{l} \in \text{lights}} (\mathbf{l} \cdot \mathbf{n}_{\mathbf{p}})\, i_d \tag{1}$$

where k_a is an ambient reflection coefficient, i_a is the ambient color of the material, $\mathbf{l}$ is the light direction, $\mathbf{n}_{\mathbf{p}}$ is the surface normal at point $\mathbf{p}$ and i_d is the diffuse color. In this setting two meshes will result in identical renderings if for each point (or pixel) on the view plane the angle between the corresponding normal on the surface and the given light direction is identical. Belhumeur et al. [1999] defined this as the "Bas-Relief Ambiguity" for the orthographic case and formulated a set of invariant transformations of the surface geometry and the corresponding light sources. In real world scenarios the lighting direction $\mathbf{l}$ is often not known in advance and difficult to control (e.g. the sunlight). Therefore, if an object is confined to a smaller space, we are looking for a deformation of the geometry which tries to preserve the surface normals to minimize the visual difference under various illumination conditions.

View-dependent surface similarity. We chose to constrain each point $\mathbf{p}'$ of the deformed surface $\mathcal{S}$ to stay on the ray emanating from a viewpoint $\mathbf{o}$ in the direction of $\mathbf{p}$ (see inset). This representation naturally preserves the surface normals under uniform scaling of the geometry for a fixed perspective. It allows us to define a surface similarity $d(\mathcal{S}, \mathcal{S}^0, \mathbf{o})$ to measure the perceived difference of the surface $\mathcal{S}^0$ and its deformed state $\mathcal{S}$ when observed from a fixed viewpoint $\mathbf{o}$:



$$d(\mathcal{S}, \mathcal{S}^0, \mathbf{o}) = \int_{\mathcal{S}} \left\| \mathbf{n}^{\mathcal{S}}_{\phi(\mathbf{p},\mathbf{o})} - \mathbf{n}^{\mathcal{S}^0}_{\mathbf{p}} \right\|^2 d\mathbf{p}. \tag{2}$$

Here, $\phi(\mathbf{p}, \mathbf{o})$ denotes the pointwise identification of the surface $\mathcal{S}^0$ with its deformed version $\mathcal{S}$. Note that we integrate over $\mathcal{S}$ to incorporate the change in the deformed surface area. In this work, we use a variational approach to compute an appearance-mimicking surface that minimizes the distance d, given user-provided thickness constraints.

Surface discretization. We represent the surface $\mathcal{S}$ as a triangle mesh $\mathcal{M} = \{\mathbf{V}, \mathcal{F}\}$, where $\mathbf{V}$ is an n-by-3 matrix that stores the coordinates of the vertices and $\mathcal{F}$ is an m-by-3 matrix encoding the connectivity. We can equivalently represent the ith vertex $\mathbf{v}_i$ of $\mathcal{S}$ as:

$$\mathbf{v}_i = \mathbf{o} + \|\mathbf{v}_i - \mathbf{o}\| \frac{\mathbf{v}_i - \mathbf{o}}{\|\mathbf{v}_i - \mathbf{o}\|} = \mathbf{o} + \lambda_i \frac{\mathbf{v}_i - \mathbf{o}}{\|\mathbf{v}_i - \mathbf{o}\|}, \tag{3}$$

where $\lambda_i = \|\mathbf{v}_i - \mathbf{o}\|$. Without loss of generality, we can assume that $\mathbf{o} = (0, 0, 0)$ and simplify Eq. (3):

$$\mathbf{v}_i = \lambda_i \hat{\mathbf{v}}_i, \quad \text{where } \hat{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{v}_i\|. \tag{4}$$

Fixing the directions $\hat{\mathbf{v}}_i$, the positions of the vertices of $\mathcal{M}$ can be expressed as a vector $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, ..., \lambda_n\}$. In this representation, all vertices defined by a choice of λ_i will project to $\mathbf{v}_i$ if seen from the viewpoint $\mathbf{o}$. Expressing $\mathcal{M}$ and $\mathcal{M}^0$ in the same parametrization, i.e. if both of them are represented as a set of some λ_i, Eq. (2) can be discretized as:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i \|(\mathbf{n}_i - \mathbf{n}_i^0)\|^2, \tag{5}$$

where A_i is the Voronoi area associated with the ith vertex.

Linearization. We can write the surface normals as a function of the vertex positions using the discrete Laplace-Beltrami operator:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i \left\| \frac{(\mathbf{L}\mathbf{D}_{\boldsymbol{\lambda}}\hat{\mathbf{V}})_i}{H_i} - \frac{(\mathbf{L}^0\mathbf{D}_{\boldsymbol{\lambda}^0}\hat{\mathbf{V}})_i}{H_i^0} \right\|^2, \tag{6}$$

where $\mathbf{L}, \mathbf{L}^0$ are discrete Laplace-Beltrami operators of $\mathcal{M}, \mathcal{M}^0$ and H_i, H_i^0 are the discrete mean curvatures at vertex i of $\mathcal{M}, \mathcal{M}^0$, respectively; $\hat{\mathbf{V}}$ is the n-by-3 matrix stacking all $\hat{\mathbf{v}}_i$s and $\mathbf{D}_{\boldsymbol{\lambda}}$ is a diagonal matrix with entries λ_i on the diagonal (and similarly for $\mathbf{D}_{\boldsymbol{\lambda}^0}$). The notation $(*)_i$ means that we extract the ith row of $*$. For more details about the definition of the discrete Laplace-Beltrami operator and the mean curvature we refer the reader to [Botsch et al. 2010].

Similarly to previous deformation algorithms [Botsch and Sorkine 2008], we linearize this expression by replacing the area weighting and the Laplacian of the deformed mesh $\mathcal{M}$ with the corresponding quantities and operators of the original mesh $\mathcal{M}^0$.

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i^0 \left\| \frac{(\mathbf{L}^0\mathbf{D}_{\boldsymbol{\lambda}}\hat{\mathbf{V}})_i}{H_i^0} - \frac{(\mathbf{L}^0\mathbf{D}_{\boldsymbol{\lambda}^0}\hat{\mathbf{V}})_i}{H_i^0} \cdot \frac{\lambda_i}{\lambda_i^0} \right\|^2. \tag{7}$$

The scaling factor λ_i/λ_i^0 compensates for the change in scale of the Laplacian vector due to the linearization (see Figure 5). Introducing this factor makes d invariant to uniform scaling of $\mathcal{M}$, similarly to the scale-invariant Laplacian deformation energy proposed in [Sorkine et al. 2004]. The main difference is that in our parametrization, the local scaling is given in closed form, as opposed to the local least-squares fitting of [Sorkine et al. 2004]. In our case, scaling λ_i induces a uniform scaling in the neighborhood of λ_i (see Eq. (4)). The scale invariance introduces rank deficiency in the optimization, but can be fixed by constraining the λ_i of a single vertex i. This can be modeled as an equality constraint:

$$\mathbf{C}_E\,\boldsymbol{\lambda} = \mathbf{b}. \tag{8}$$

Linear approximation effect. The linearization error introduced in Eq. (7) is higher in areas with low curvature, as shown in Figure 6, where the depth of the *Box* model is constrained to a small range. To compare, we iteratively computed the more accurate solution of the nonlinear problem, updating the cotangent Laplacian of the deformed surface in every iteration. Unfortunately, this approach works only for very small and regular meshes and does not converge for any other result shown in the paper. The reason is that the cotangent approximation of the Laplacian becomes increasingly inexact for triangles with angles exceeding $90°$. Therefore, we opt for using the less accurate but much more stable and efficient linearization in Eq. (7).

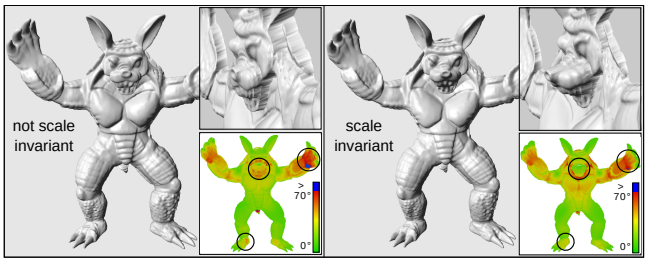


Figure 5: *Using a scale-dependent Laplacian (i.e., omitting the term λ_i/λ_i^0 in Eq. (7)) introduces artifacts (left) that disappear when using our formulation (right). The colored insets visualize the angular difference of the normals between the deformed and the initial surface.*

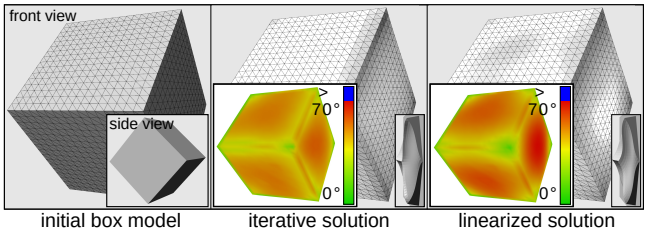


Figure 6: *The depth of the* Box *model (1538 vertices) is constrained to a small range. The angular difference of the normals (color insets) introduced by the linearization in Eq. (7) (right) is higher for low-curvature regions than in the more accurate solution computed by iteratively updating the cotangent Laplacian (middle).*

Bias for high frequency details. Eq. (7) is challenging to minimize numerically, due to the extreme variation in the range of the term H_i^0. The curvature is close to zero in all flat or very smooth parts of the mesh, introducing an extreme scaling, which leads to numerical problems. We remove this instability by adding a weighting that biases the error measure towards preserving high-curvature details. We weigh the norm of vertex i with $(H_i^0)^2$, hence giving less importance to the unstable flat regions and canceling out H_i^0:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i^0 \left\| (\mathbf{L}^0\mathbf{D}_{\boldsymbol{\lambda}}\hat{\mathbf{V}})_i - (\mathbf{L}^0\mathbf{D}_{\boldsymbol{\lambda}^0}\hat{\mathbf{V}})_i \frac{\lambda_i}{\lambda_i^0} \right\|^2 \tag{9}$$

The effect of introducing the bias is minor, as shown in Figure 7, but makes the optimization numerically stable.

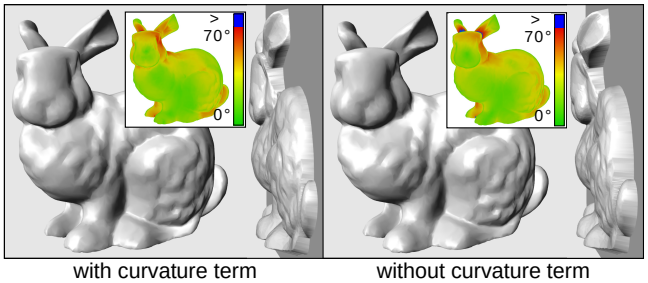


Figure 7: *Ignoring the mean curvature term has a minimal effect on the results and makes the optimization numerically stable. The insets show the angular difference of the normals between the deformed and the initial surface.*

Depth constraints. Eq. (9) is quadratic in $\boldsymbol{\lambda}$ and can be efficiently minimized by solving a linear system. The thickness of the resulting surface is completely controlled by $\boldsymbol{\lambda}$, which can be easily bounded on each vertex using inequality constraints of the form:

$$\lambda_i^{\min} \leq \lambda_i \leq \lambda_i^{\max}. \tag{10}$$

This transforms the minimization of Eq. (9) into a quadratic problem, which can still be optimally solved.

Disconnected pieces. Depending on the application, it might be useful to define depth range constraints that are discontinuous. This would enable us to optimize for appearance-mimicking surfaces that are themselves discontinuous, like the pillar surfaces in Figure 15. This could be achieved by splitting the mesh into multiple disconnected sets of vertices and solving independent optimizations. However, this approach requires remeshing and splitting the shared boundary between every group of vertices, potentially generating low-quality triangles that make any further optimization unstable. Moreover, additional constraints to match the normals for shared vertices are then needed.

We therefore propose a different approach to directly use our algorithm, without the need to split the mesh. Instead of providing absolute lower and upper bounds $\lambda_i^{\min}$ and $\lambda_i^{\max}$ for each vertex, we define a dynamic range for each group of independent vertices, which can be freely moved during the optimization. This provides maximal freedom to optimize the energy, and only after the optimization the surface is cut into pieces and displaced according to the original, discontinuous geometry. We model this idea by adding a variable μ_g for each group g of independent vertices, transforming the constraints into:

$$\mu_g \lambda_i^{\min} \leq \lambda_i \leq \mu_g \lambda_i^{\max}. \tag{11}$$

We sketch an example in Figure 8, where the depth bounds are independently provided for each disconnected pillar (left). Our optimization finds the optimal appearance-mimicking surface (AMS) that satisfies the depth bounds up to a scaling, which is controlled by the μs. After the optimization, the λs are scaled back by multiplying by the inverse of the μs to warp each piece of the AMS back to its associated pillar (right). The point $\mathbf{p}$ (in blue) is fixed to make the solution unique, as discussed previously. The result, after scaling back, is independent of the choice of $\mathbf{p}$. Note that the same formulation with only one λ is used for the case of one simple continuous depth range constraint.

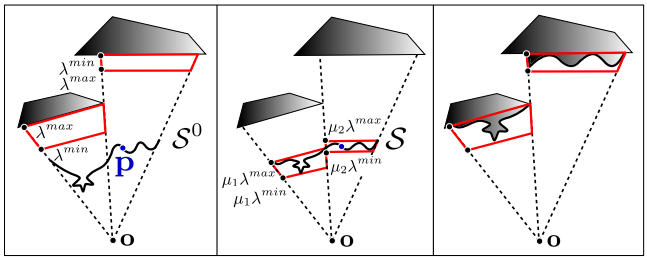
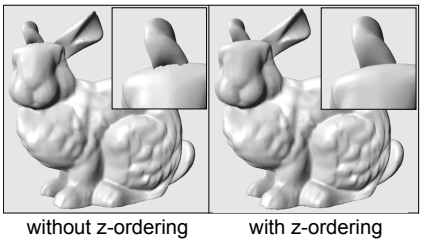


Figure 8: *Depth constraints can be specified independently for every disconnected component of the target surface (left). The bounds are moved during the optimization to enlarge the solution space and increase the AMS quality (middle). The optimized surface is then projected to every component, guaranteeing to exactly satisfy the original depth bounds.*

Self-intersection avoidance. The deformed surface might contain self-intersections, as visible in the boundary between the head and the ear of the *Bunny* relief (see inset). Since each vertex is constrained to move on a ray, this can only happen when two parts of the surface change their depth ordering, with respect to the viewpoint $\mathbf{o}$. To prevent self-intersections, we force vertices to preserve their depth relationships: for every vertex $\mathbf{v}_i^0$ we cast a ray from $\mathbf{o}$ in direction $\hat{\mathbf{v}}_i^0$. For every pair of consecutive hits, we add a linear inequality constraint that enforces depth ordering preservation. Note that the rays will generally hit the interior of a triangle, and the hit position can then be represented using barycentric coordinates. All inequalities can be stacked in matrix form as:



$$\mathbf{C}_I \boldsymbol{\lambda} \leq \mathbf{d}. \tag{12}$$

While this procedure does not guarantee the elimination of edge-to-edge intersections, we found that this is not a problem in our experiments, and it does not affect the 3D-printed results since the resolution of the printer is typically lower than the mesh resolution.

Height fields. Our formulation can be specialized to create appearance-mimicking surfaces that are height fields w.r.t. the viewpoint $\mathbf{o}$, thereby increasing the optimization efficiency and quality of the results, especially for very thin bas-reliefs and carvings.

Assume we wish to create a height field AMS from a general surface $\mathcal{S}$. Many parts will not be visible from $\mathbf{o}$ due to self occlusion: Constraining such vertices would unnecessarily restrict the degrees of freedom in the optimization, since the occluded vertices will not be visible. Therefore, we only set hard constraints on the visible vertices, which considerably speeds up the optimization (for the *Bunny* model used in Figure 10 the computation time is reduced by a factor of 3.6). If large regions of a surface are occluded, the quality can be further increased by dampening the influence of the corresponding vertices in the energy, giving more freedom to the visible parts. To model this optional feature, we introduce an additional weighting w_i that scales the difference in the vertex normals:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} w_i^2 A_i^0 \left\| (\mathbf{L}^0 \mathbf{D}_{\boldsymbol{\lambda}} \hat{\mathbf{V}})_i - (\mathbf{L}^0 \mathbf{D}_{\boldsymbol{\lambda}^0} \hat{\mathbf{V}})_i \frac{\lambda_i}{\lambda_i^0} \right\|^2. \tag{13}$$

In Figure 9, we assign a weight of 1 to the visible vertices and 0.1

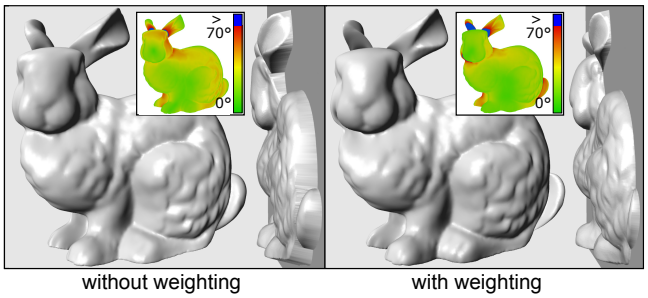


Figure 9: *By dampening the influence of the hidden vertices with the weights w_i, we leave more freedom to the optimization, which better preserves the surface details in the visible regions.*

to the others, obtaining a better approximation of the geometric details. We enabled this additional weighting only for the surface in Figures 9 and 11. The weighting could be exposed to the users, enabling to easily control the deformation by specifying which parts are important to preserve.

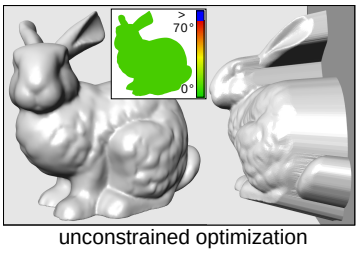
Optimization. The energy and the constraints can be written in matrix form:

$$\begin{aligned} \underset{\boldsymbol{\lambda},\boldsymbol{\mu}}{\text{minimize}} \quad & \|\mathbf{D}_A\,\mathbf{D}_w(\tilde{\mathbf{L}}^0\mathbf{D}_{\hat{\mathbf{V}}} - \mathbf{D}_{\mathbf{L}_\theta})\,\mathbf{S}\,\boldsymbol{\lambda}\|^2 + \alpha\|\boldsymbol{\mu}\|^2 \\ \text{subject to} \quad & \mathbf{C}_I\,[\boldsymbol{\lambda}\ \ \boldsymbol{\mu}]^\intercal \leq \mathbf{d}, \\ & \mathbf{C}_E\,[\boldsymbol{\lambda}\ \ \boldsymbol{\mu}]^\intercal = \mathbf{b} \end{aligned} \tag{14}$$

Where $\mathbf{D}_A$ and $\mathbf{D}_w$ are $3n$-by-$3n$ diagonal matrices containing the square roots of the areas A_i^0 and the weights w_i, respectively. $\tilde{\mathbf{L}}^0$ is a $3n$-by-$3n$ matrix and equal to $\mathbf{L}^0 \otimes \mathbf{I}_3$, where $\otimes$ is the Kronecker product, $\mathbf{D}_{\hat{\mathbf{V}}}$ is a $3n$-by-$3n$ diagonal matrix of the row-wise stacked elements of $\hat{\mathbf{V}}$, $\mathbf{S}$ is a $3n$-by-n selector matrix, coupling all λs with the x, y, z coordinates in the system and can be written as $\mathbf{I}_n \otimes [1,1,1]^T$, $\mathbf{I}_n$ being an n-by-n identity matrix. $\mathbf{L}_\theta$ is a $3n$-vector defined as follows:

$$\mathbf{L}_\theta = \mathbf{D}^{-1}_{(\mathbf{S}\boldsymbol{\lambda}^0)}\,\tilde{\mathbf{L}}^0\,\mathbf{D}_{\hat{\mathbf{V}}}\,\mathbf{S}\,\boldsymbol{\lambda}^0. \tag{15}$$

The regularization term on μ is necessary to make the energy gradient matrix full-rank. We set $\alpha = 10^{-7}$ in all our experiments. This regularization has an intuitive meaning: it makes the minimizer unique by selecting the smallest μs which correspond to moving the surface as close as possible to the upper end of the depth range constraint. Note that this is indeed the desired behavior, since it minimizes the thickness of the AMS on each disconnected part. The influence of the regularization can be neglected and the solution of an unconstrained optimization is identical to the initial model up to numerical errors, as shown in the inset.



unconstrained optimization

We experimentally discovered that converting our QP formulation to the equivalent conic program greatly improves performance, on average by a factor of 5. We use the multi-threaded conic solver in MOSEK for all our experiments [Andersen and Andersen 2000]. See Appendix for the implementation details.

	Interactive			Final		
Model	**#V**	**#F**	**Time**	**#V**	**#F**	**Time**
Dragon	41k	83k	51s	300k	600k	1297s
Owl	20k	41k	32s	218k	437k	760s
Dragon Head	20k	39k	28s	304k	609k	825s
Armadillo	10k	20k	28s	180k	361k	2961s
Fish	10k	19k	11s	279k	559k	920s
Bunny	10k	19k	10s	40k	79k	85s
Face	10k	19k	9s	40k	80k	42s
Cow Herd	24k	51k	9s	417k	834k	330s
Pillar Forest	22k	43k	55s	221k	444k	435s

Table 1: *Statistic of all the used model meshes and the times needed for the conic optimization.*

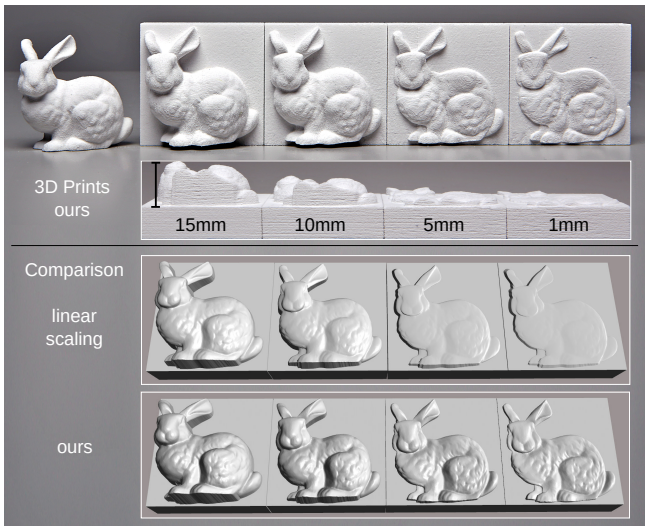


Figure 10: *A sequence of height restricted bas-reliefs from left to right using the* Bunny *model. Top: 3D prints generated by our method. Bottom: comparison between linear scaling and our method.*

4 Results

We used a quad-core Intel i7 processor clocked at 3.4 GHz to compute all our results. Our prototype is written in C++/MATLAB and uses the MOSEK solver for the conic optimization [Andersen and Andersen 2000]. Statistics on the meshes and on the computation times are summarized in Table 1. To support interactive design of our examples, we used a low-resolution version of each model (left part). We used a ZCorp 650 to 3D print our results, employing a clear binder color and default printing options.

Variable depth. We compare our method with a simple linear compression in a sequence of bas-reliefs ordered according to decreasing depth. As can be seen in Figure 10, our approach better preserves the geometric details even for extremely thin surfaces.

Comparison with [Weyrich et al. 2007]. We compare our results with [Weyrich et al. 2007] in Figure 11, using the same model and viewpoint. We reimplemented the method of [Weyrich et al. 2007] without the optional post-processing sharpening step and used the same triangle based discretization for both methods. The two results share many similarities, with a slight edge for our method that better preserves the fine details. One important difference is that our algorithm allows to exactly control the depth of the bas-relief

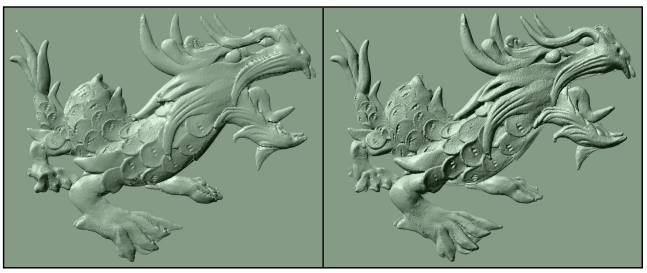

Weyrich et al. 2007 ours

Figure 11: *Comparison of a Dragon bas-relief with [Weyrich et al. 2007].*

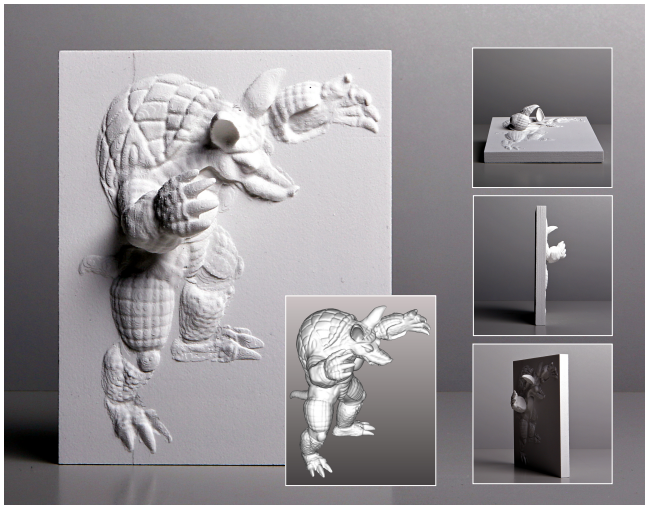


Figure 12: *A non-height field* Armadillo *relief, where the arm and one ear are left unconstrained, creating an interesting effect when observed from different angles.*

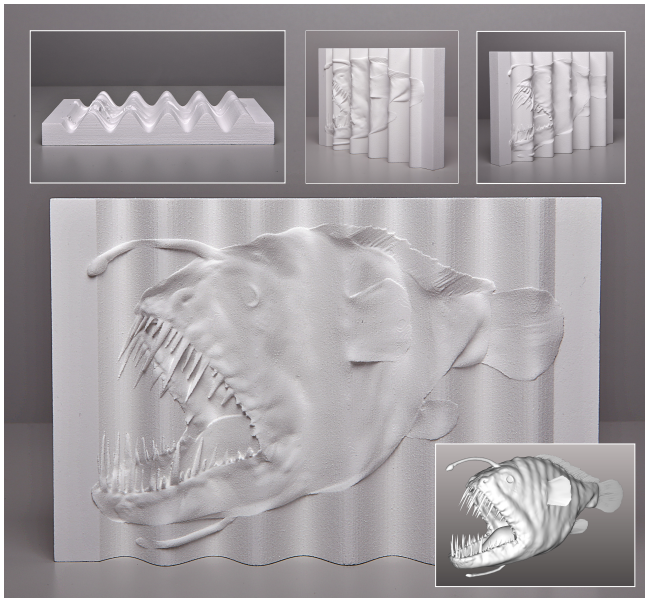


Figure 13: *A fish embedded in the wavy surface of the sea, constrained to stay within a 4 mm thin layer.*

without resorting to a linear scaling in the post-processing. For a physically-printed model of size 24 cm $\times$ 19 cm, the bas-reliefs protrude by only 8 mm from the baseplate.

Non-height field bas-reliefs. In contrast to existing methods, our AMSs do not need to be height fields. In Figure 12 we show a depth-compressed *Armadillo*, where we constrain only the visual hull vertices to exactly map to the surface of the baseplate, while letting the arm and one ear unconstrained. The transition between the two regions is not a height field from the chosen viewpoint, creating an interesting effect when observing from different angles.

3D Camouflage. Our method gives us exact control over the depth of each vertex. By exploiting this feature, we can constrain the AMSs to lie on many different and disconnected surfaces, as discussed in Section 3 and shown in Figure 15. We project four models from four different viewpoints onto a "forest" of pillars. While it is impossible to discern any pattern when observing from an arbitrary viewpoint, the models reveal themselves if viewed from one of the four special viewpoints. It is not trivial to manually construct such arragments of pillars, as the projected models should not interfere with the views from other viewpoints. Therefore, we implemented a simple editor which allows us to specify multiple viewpoints and pillar shapes and constantly visualizes the visible range for each of the four viewpoints in different colors as shown in Figure 14. The necessary depth range constraints are then automatically extracted and they guarantee intersection free views of all models within the complex "pillar forest".

Inspired by street artwork painted over the steps of a staircase, we use our algorithm to embed a 3D model of an owl into a staircase, as shown in Figure 3. Note that the AMS is very thin, so it does not affect the function of the staircase, while being much more resistant than paint to erosion and aging. This idea is very general, and we plan to investigate it further in future works, applying it to design furniture, jewelry and architecture.

Stress tests. We stress test our method with two difficult sets of constraints. In Figure 13, we constrain the *Fish* model to stay within a 4 mm thin layer of a wavy surface mimicking the sea. The constraints strongly restrict the deformation, but our algorithm is still able to use the available space to generate a high-quality relief. The result is surprisingly close to the original model from the desired viewpoint, but the illusion immediately dissipates when the model is rotated.

Carving bas-reliefs. Automatically generated bas-reliefs have mostly been created on flat or simple surfaces by previous algorithms. Our method can handle very complex cases without any modification, as we show in Figure 4. We carve a *Dragon Head* inside a V-shaped volume, constraining the surface to carve up to

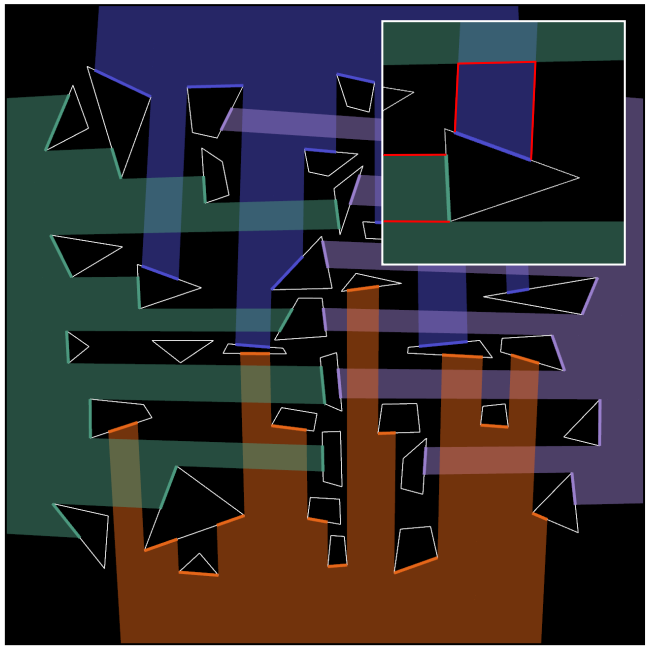


Figure 14: *Top down visualization of the "pillar forest" model (Figure 15) in our visualizer. Each color corresponds to the visibility range of a viewpoint. The inset shows the constrained spaces (red lines) for the fragmented model parts.*

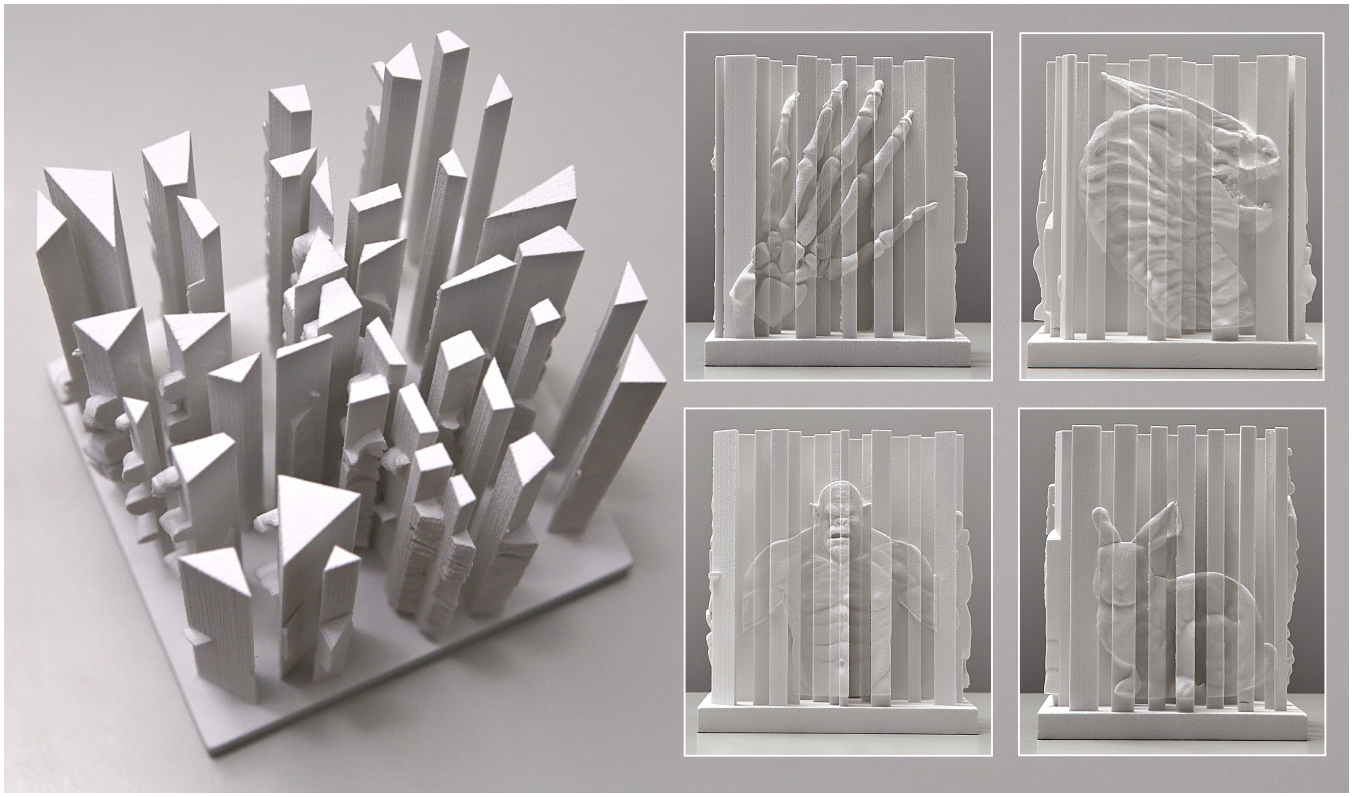


Figure 15: *Four 3D models are obfuscated in a "pillar forest". Refer to the accompanying video for a demonstration.*

6 mm in the volume and stick outside for no more than 1 mm. Both bounds are hard constraints that are guaranteed to be satisfied by our optimization.

Multiple view strips. Multiple views of the same *Head* model embedded in five planar surfaces are shown in Figure 16. Note that a highly nontrivial deformation is introduced to constrain the thickness of the different views to not exceed 2 mm.

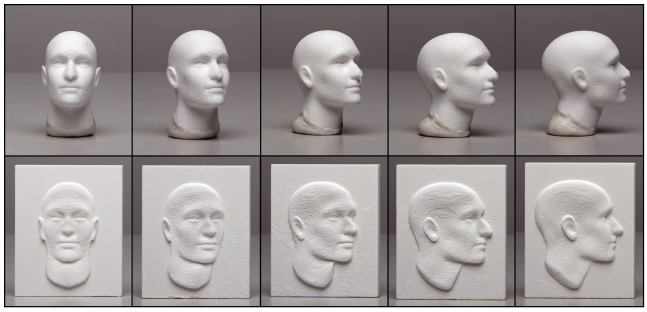


Figure 16: *Multiple views of the* Head *model (80k vertices), rotated by 22.5 degrees and constrained to a thickness of 2 mm.*

Multiple objects. It is possible to create an AMS from a 3D scene composed of multiple disconnected objects. We show an example in Figure 17, where the visual hull of every cow is constrained to lie on the baseplate.

Natural lighting conditions. We demonstrate that our simplified lighting model (Section 3) is sufficient for the generation of realistic AMS in Figure 18, where we took photographs of our 3D printed results under outdoor lighting conditions. The photograph captured during a cloudy day (bottom) perfectly preserves the depth illusion, while direct exposure to sunlight (top) creates harsh shadows that reveal the extreme thinness of the *Fish* and the *Armadillo* reliefs.



Figure 17: *A 3D scene containing a small cow family is converted to a thin relief. The inset shows a rendering of the original scene from the top view.*



Figure 18: *A collection of appearance-mimicking surfaces photographed in a courtyard during a sunny day (top) and a cloudy day (bottom).*

5 Concluding remarks

We presented a novel approach to create appearance-mimicking surfaces, which uses a special mesh parameterization of the deformation to robustly optimize for a surface whose normals are similar to the input geometry from a fixed perspective. Our algorithm supports exact and adaptive depth constraints and works directly on manifold triangle meshes without the need for resampling. We demonstrate the effectiveness of our approach to generate bas-reliefs and artistic compositions. Our lighting model assumes diffuse material and directional lighting and it does not account for self-shadowing. If these conditions are far from being satisfied, it produces sub-optimal results: We show a failure case in Figure 19, where a flash gun is placed close to the model and pointed directly at it. The harsh lighting creates specular reflections and strong shadows which ruin the illusion. We plan to investigate the use of a more accurate lighting model in future work.

The illusion generated by our method degrades as the viewpoint gets closer to the object, since the stereo disparity increases, helping the human vision system to detect the illusion. This problem has not been explored in the literature, and, given the quick development of 3D displays, is an interesting venue for future work.

With the advent of commodity 3D printing, we expect that bas-reliefs will be widely used to personalize objects and wearables. Our algorithm provides a robust and efficient way to support non-expert users in effectively using bas-reliefs in their creations.

Acknowledgements

The authors thank the anonymous reviewers for their helpful comments and suggestions. Some of the models shown in the paper were provided by AIM@SHAPE and the Stanford Computer Graphics Laboratory. This work was supported in part by the ERC grant iModel (StG-2012-306877).



Figure 19: *Under harsh lighting conditions the depth illusion does not work. This photograph has been taken with one single off-camera strobe casting light from left to right.*

References

ALEXA, M., AND MATUSIK, W. 2010. Reliefs as images. *ACM Trans. Graph. 29*, 4.

ANDERSEN, E. D., AND ANDERSEN, K. D. 2000. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High Performance Optimization*. Kluwer Academic Publishers, 197–232. `http://www.mosek.com`.

BARAN, I., KELLER, P., BRADLEY, D., COROS, S., JAROSZ, W., NOWROUZEZAHRAI, D., AND GROSS, M. 2012. Manufacturing layered attenuators for multiple prescribed shadow images. *Comput. Graph. Forum 31*, 2, 603–610.

BELHUMEUR, P. N., KRIEGMAN, D. J., AND YUILLE, A. L. 1999. The bas-relief ambiguity. *International Journal of Computer Vision 35*, 1, 33–44.

BERMANO, A., BARAN, I., ALEXA, M., AND MATUSIK, W. 2012. Shadowpix: Multiple images from self shadowing. *Comput. Graph. Forum 31*, 2, 593–602.

BIAN, Z., AND HU, S.-M. 2011. Preserving detailed features in digital bas-relief making. *Computer Aided Geometric Design 28*, 4, 245–256.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graph. 14*, 1, 213–230.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LÉVY, B. 2010. *Polygon Mesh Processing*. A K Peters.

CHANDRAKER, M. K., KAHL, F., AND KRIEGMAN, D. J. 2005. Reflections on the generalized bas-relief ambiguity. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, 788–795.

CHU, H.-K., HSU, W.-H., MITRA, N. J., COHEN-OR, D., WONG, T.-T., AND LEE, T.-Y. 2010. Camouflage images. *ACM Trans. Graph. 29*, 3.

CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 1997. Computer-assisted generation of bas-and high-reliefs. *J. Graphics, GPU, & Game Tools 2*, 3, 15–28.

ELBER, G. 2010. Ortho-pictures: 3D objects from independent 2D data sets. In *Advances in Architectural Geometry 2010*, C. Ceccato et al., Eds. Springer, 175–192.

JI, Z., MA, W., AND SUN, X. 2014. Bas-relief modeling from normal images with intuitive styles. *IEEE Trans. Vis. Comput. Graph. 20*, 5, 675–685.

KERBER, J., TEVS, A., BELYAEV, A. G., ZAYER, R., AND SEIDEL, H.-P. 2009. Feature sensitive bas relief generation. In *Shape Modeling International*, 148–154.

KOLOMENKIN, M., LEIFMAN, G., SHIMSHONI, I., AND TAL, A. 2011. Reconstruction of relief objects from line drawings. In *Proc. CVPR*, 993–1000.

MITRA, N. J., AND PAULY, M. 2009. Shadow art. *ACM Trans. Graph. 28*, 5.

SELA, G., AND ELBER, G. 2007. Generation of view dependent models using free form deformation. *The Visual Computer 23*, 3, 219–229.

SONG, W., BELYAEV, A. G., AND SEIDEL, H.-P. 2007. Automatic generation of bas-reliefs from 3d shapes. In *Shape Modeling International*, 211–214.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proc. Symposium on Geometry Processing*, 175–184.

SUN, X., ROSIN, P. L., MARTIN, R. R., AND LANGBEIN, F. C. 2009. Bas-relief generation using adaptive histogram equalization. *IEEE Trans. Vis. Comput. Graph. 15*, 4, 642–653.

TAN, P., QUAN, L., AND ZICKLER, T. 2011. The geometry of reflectance symmetries. *IEEE Trans. Pattern Anal. Mach. Intell. 33*, 12, 2506–2520.

WEYRICH, T., DENG, J., BARNES, C., RUSINKIEWICZ, S., AND FINKELSTEIN, A. 2007. Digital bas-relief from 3D scenes. *ACM Trans. Graph. 26*, 3, 32.

ZATZARINNI, R., TAL, A., AND SHAMIR, A. 2009. Relief analysis and extraction. *ACM Trans. Graph. 28*, 5.

ZHANG, Y.-W., ZHOU, Y.-Q., ZHAO, X.-F., AND YU, G. 2013. Real-time bas-relief generation from a 3D mesh. *Graphical Models 75*, 1, 2–9.

Appendix

We use MOSEK [Andersen and Andersen 2000] to efficiently solve sparse, quadratic programming problems. Its documentation strongly recommends converting convex quadratic energy minimization with linear inequality constraints, like Eq. (14), into linear energy minimization with conic constraints. We found this to be especially advantageous for our problem, which is of the form:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{F}\mathbf{x}\|^2 + \mathbf{f}^\intercal\mathbf{x} + \text{const} \\ \text{subject to} \quad & \mathbf{C}_I\,\mathbf{x} \le \mathbf{d}, \;\; \mathbf{C}_E\,\mathbf{x} = \mathbf{b} \end{aligned} \tag{16}$$

with

$$\begin{aligned} &\mathbf{x} = [\boldsymbol{\lambda}\ \boldsymbol{\mu}]^\intercal \\ &\mathbf{f} = \mathbf{0} \\ &\mathbf{F} = \begin{bmatrix} \mathbf{D}_A\mathbf{D}_w(\tilde{\mathbf{L}}^0\mathbf{D}_{\tilde{\mathbf{V}}} - \mathbf{D}_{\mathbf{L}_\theta})\mathbf{S} & \mathbf{0} \\ \mathbf{0} & \sqrt{\alpha}\cdot\mathbf{I} \end{bmatrix}. \end{aligned}$$

The matrix $\mathbf{I}$ is an identity matrix of size of the length of $\boldsymbol{\mu}$. To convert this to a conic problem, we first introduce a vector of auxiliary variables $\mathbf{t}$ and rewrite Eq. (16) as:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{t}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{t}\|^2 + \mathbf{f}^\intercal\mathbf{x} + \text{const} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \le \mathbf{d}, \;\; \mathbf{C}_E\,\mathbf{x} = \mathbf{b}, \\ & \mathbf{F}\,\mathbf{x} - \mathbf{t} = \mathbf{0}. \end{aligned}$$

Using the scalar variables v and c we convert into conic form:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{t},v,c}{\text{minimize}} \quad & v + \mathbf{f}^\intercal\mathbf{x} + \text{const} \\ \text{subject to} \quad & \mathbf{C}_I\,\mathbf{x} < \mathbf{d}, \;\; \mathbf{C}_E\,\mathbf{x} = \mathbf{b}, \\ & \mathbf{F}\,\mathbf{x} - \mathbf{t} = \mathbf{0}, \\ & cv \ge \sum_i t_i^2, \;\; c = 2, \;\; v \ge 0, \end{aligned} \tag{17}$$

where the inequality constraint on v forces its value to be inside the cone described by the coordinates of $\mathbf{t}$. Putting all variables in a column vector, we can write this in matrix form, as we supply it to the solver:

$$\begin{aligned} \underset{[\mathbf{x}^\intercal\ \mathbf{t}^\intercal\ v\ c]}{\text{minimize}} \quad & \begin{bmatrix} \mathbf{f}^\intercal & \mathbf{0}^\intercal & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} + \text{const} \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{F} & -\mathbf{I} & 0 & 0 \\ \mathbf{C}_I & \mathbf{0}^\intercal & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \ge \begin{bmatrix} 0 \\ -\infty \end{bmatrix} \\ & \begin{bmatrix} \mathbf{F} & -\mathbf{I} & 0 & 0 \\ \mathbf{C}_I & \mathbf{0}^\intercal & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \le \begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix} \\ & \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \le \begin{bmatrix} \mathbf{b} \\ \infty \\ \infty \\ 2 \end{bmatrix}, \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \ge \begin{bmatrix} \mathbf{b} \\ -\infty \\ 0 \\ 2 \end{bmatrix} \\ & cv \ge \sum_i t_i^2. \end{aligned}$$

Note that every equality constraint was replaced by two inequality constraints. Because in our case $\mathbf{C}_E$ is a diagonal matrix, we can represent these equality constraints with an upper and lower bound on $\boldsymbol{\lambda}$, which can be handled more efficiently in the optimization.