Fall 2017 CSE528 Final Project Technical Report: Feature Sensitive Bas-Relief Generation

Abiyaz Chowdhury

12/7/2017

Abstract

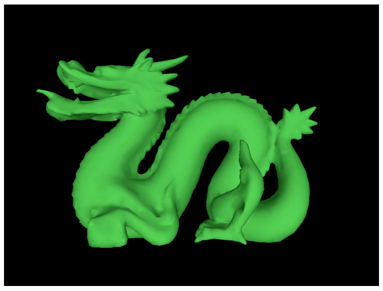
This project implements an improved technique for bas relief generation from the depth maps of 3D models. In particular, the implementation is designed to capture high level surface detail that would otherwise be lost during the depth compression process in other techniques for relief generation. The technique is essentially based on bilinear filtering [**?**] in the gradient domain, and aims to achieve a strong balance of good compression as well as preservation of surface details.

1 Introduction

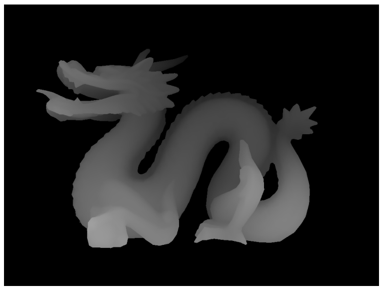
Bas relief is a type of sculpture where the sculpture remains attached to a background surface, protruding very little from the surface, giving the appearance of an almost flat surface when viewed sideways. But when viewed from a direction orthogonal to the sculpture, the bas relief appears to be almost like a 3D object. Relief has been used as a popular art form spanning much of history and many different civilzations, gracing the decorum of museums, courts, palaces, and places of worship. This project discusses the automatic generation of bas-reliefs (basso relievo), where the sculpture protrudes only to low heights from the base surface. The key approach used in this project is bilinear filtering in the gradient domain, which helps to preserve sharp features that would otherwise be lost if a direct compression were ot be used.

2 Input

The input provided to the algorithm is a 3D scene. I used the Stanford green dragon model, which is freely available on http://graphics.stanford.edu/data/3Dscanrep/ in the .PLY format.



(a) Orthographic view of input 3D model



(b) Depth map

3 Depth Map

In order to generate a bas relief, the relief must be captured from an image of the model in some particular cameraspace. Therefore, we position our model so that it is centered at the origin, and capture the depth buffer of the scene when it is rendered from a camera located at (0,0,5) and starting into the negative z-axis. This is done using OpenGL's Frame Buffer Object (FBO) and then reading the contents of the buffer into an array. Much of the subsequent work is then done on 2D arrays in C++.

4 Binary segmentation

Via simple thresholding, a foreground-background segmentation (binary mask) of the image is computed:

$$B(i,j) = \begin{cases} 0 & I(i,j) = \delta \\ 1 & else \end{cases}$$

One could perhaps use a more powerful segmentation algorithm, such as SLIC superpixels, although the authors do not discuss this, and for the dragon model it is not necessary.

5 Foreground-Background Normalization

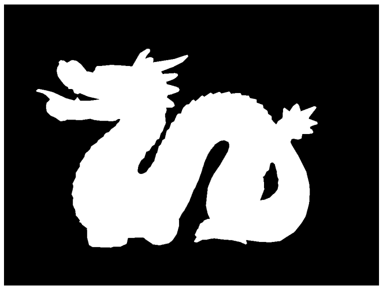
In order to ensure that the relief smoothly protrudes from the base surface, a normalization is performed so that the back of the model has the same height as the base.

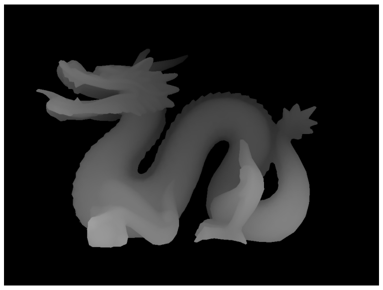
$$\hat{I}(x,y) = B(i,j)(I(x,y) - I_{min})$$

6 Image gradients

Then, both image gradients and binary mask gradients are computed:

$$\hat{I}_x(i,j) = \hat{I}(i+1,j) - \hat{I}(i,j)$$


(a) Foreground background segmentation


(b) Normalized depth map

(a) X-gradient (with gamma correction)

(b) Y-gradient (with gamma correction)

$$\hat{I}_y(i,j) = \hat{I}(i,j+1) - \hat{I}(i,j)$$
$$\hat{B}_x(i,j) = \hat{B}(i+1,j) - \hat{B}(i,j)$$
$$\hat{B}_y(i,j) = \hat{B}(i,j+1) - \hat{B}(i,j)$$

7 Computation of the boundary

Then, the foreground/background boundary is computed:

$$S(i,j) = \begin{cases} 0 & |B_x(i,j)| = 1 \ or \ |B_y(i,j)| = 1 \\ 1 & else \end{cases}$$

8 Remove boundary from image gradients

Then the image boundary is removed from the image derivatives:

$$I'_k = S \odot I_k$$

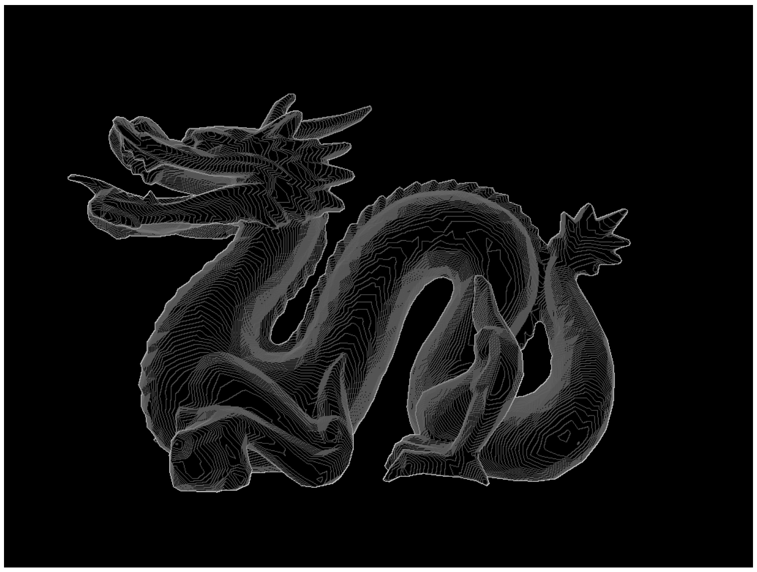
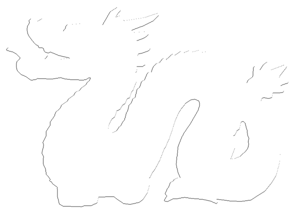


Figure 4: Gradient magnitude (with gamma correction)



(a) Boundary mask

9 Outlier detection and removal from image gradients

Then outliers are removed, (where t is a tunable parameter, values around 0.6 tended to work fairly well):

$$O(i,j) = \begin{cases} 0 & |I'_k(i,j) - \mu_k| > t * \sigma_k \\ 1 & else \end{cases}$$

$$I''_k = O \odot I'_k$$

10 Compression

Then, a compression function is used to smooth the gradient image (technically it is not a true compression function as it actually amplifies small gradients and attenuates large ones):

$$A(X,i,j) = \begin{cases} 0 & X(i,j) = 0 \\ \frac{a}{|X(i,j)|}(\frac{|X(i,j)|}{a})^b & else \end{cases}$$

$$I'''_k = A(I''_k) \odot I''_k$$

11 Bilinear Filter

Then the bilinear filter is applied:

$$BF(X,i,j) = \frac{\sum\limits_{m,n} W^{m,n}_{i,j}(X) X_{m,n}}{\sum\limits_{m,n} W^{m,n}_{i,j}(X)}$$

$$W^{m,n}_{i,j}(X) = G_{\sigma_s}(\| \begin{pmatrix} i \\ j \end{pmatrix} - \begin{pmatrix} m \\ n \end{pmatrix} \|) G_{\sigma_r}(|X_{i,j} - X_{m,n,}|)$$

12 Gradient decomposition, coarsening, and recovery

Then gradient decomposition is performed, and the gradient is subsequently recovered:

$$Coarse_k = BF(I'''_k)$$

$$Fine_k = I'''_k - Coarse_k$$

$$J_k = Fine_k + r * Coarse_k$$

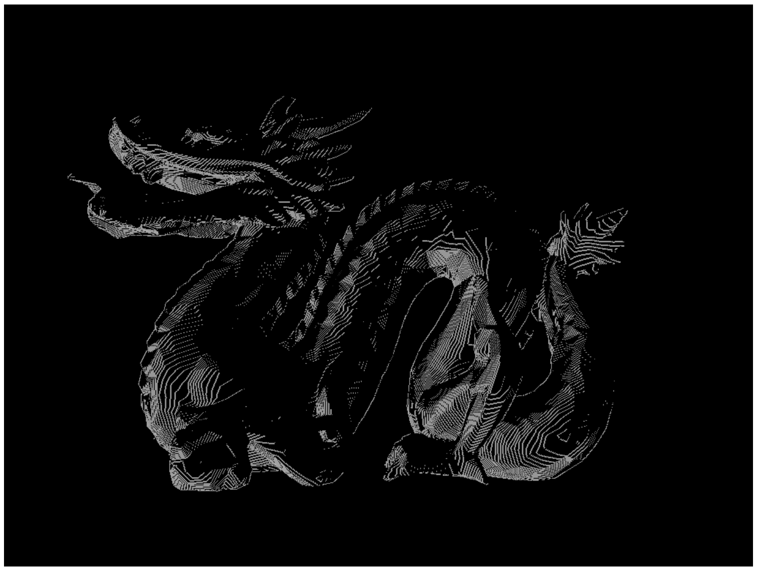
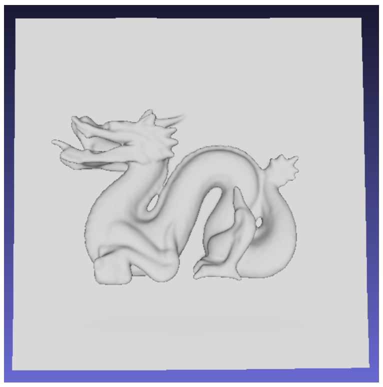
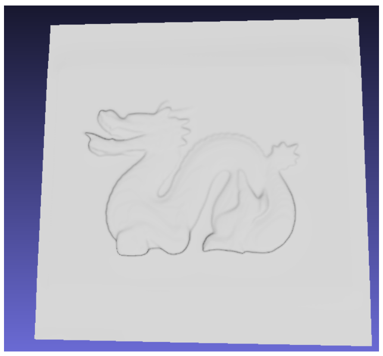


Figure 6: Laplacian


(a) Relief without any compression


(b) Relief with compression function applied

13 Computation of image Laplacian

Then the Laplacian of the gradient image is computed:

$$\Delta J = J_{xx} + J_{yy}$$

$$J_{xx}(i,j) \approx J_x(i,j) - J_x(i-1,j)$$
$$J_{yy}(i,j) \approx J_y(i,j) - J_y(i,j-1)$$

14 Poisson reconstruction

The depth map is then reconstructed by solving a Poisson equation. This is done by setting up a sparse system of linear equations, whose matrix can be shown to be diagonally dominant, and then iteratively solving using the Jacobi or the Gauss-Seidel iteration techniques. The resulting height field is then normalized and rescaled to the desired range to obtain the final bas-normalized height field:

$$\hat{J}(x,y) = B(i,j) \cdot S(i,j) \cdot (J(x,y) - J_{min})$$

$$\bar{J} = \frac{J_{bas}}{\hat{J}_{max}} \cdot \hat{J}$$

15 Triangulation and mesh generation

Once the final height field is obtained (as an .XYZ point could), a mesh needs to be generated so the relief can be visualized in 3D. Delaunay triangulation, marching cubes, and ball-pivoting surface reconstruction were all attempted but with poor results. Ultimately, screened Poisson surface reconstruction was performed, which is available on the opensource tool MeshLab, to obtain a high quality triangular mesh of the height field. The final bas reief was then exported as a .OBJ file, fully ready for rendering in OpenGL.

16 Conclusion

The final relief satisfies the height constraints that were imposed on it, as well as managing to capture some fine features of the original scene.

References

[1] Kerber, Jens, et al. *Feature sensitive bas relief generation.* Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on. IEEE, 2009.