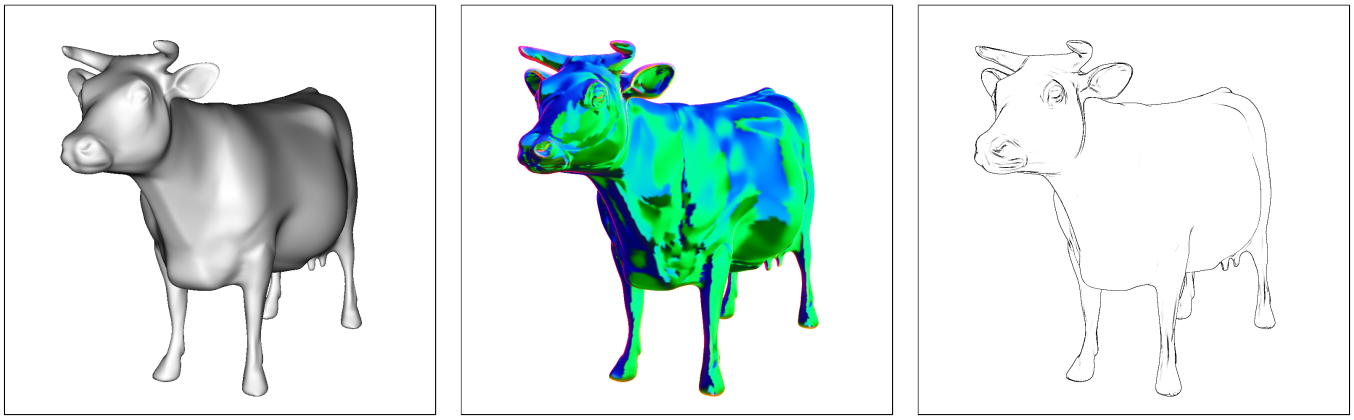
A Hybrid Method for Computing Apparent Ridges

Eric Jardim   Luiz Henrique de Figueiredo
IMPA – Instituto Nacional de Matemática Pura e Aplicada, Rio de Janeiro, Brazil
ejardim@impa.br   lhf@impa.br



shaded model   view-dependent curvature and maximum direction   apparent ridges

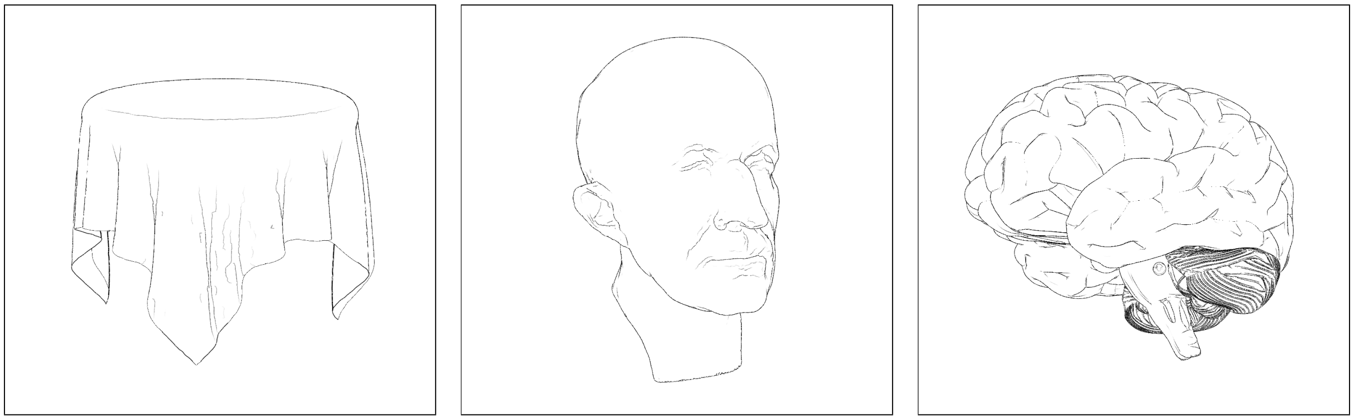
***Abstract*—We propose a hybrid method for computing apparent ridges. Our method combines object-space and image-space computations and runs partially in the GPU, taking advantage of modern graphic cards processing power and producing faster results in real time.**

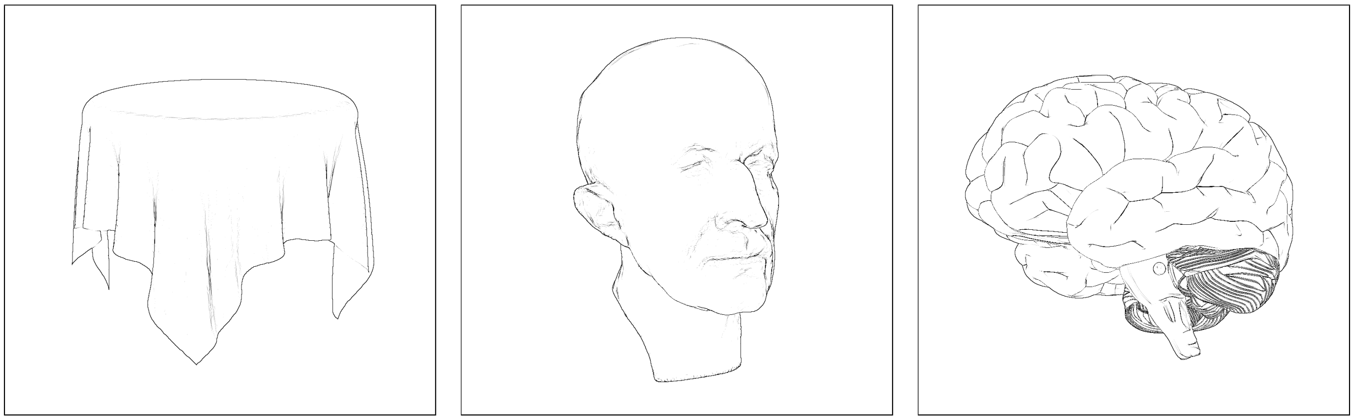
Expressive line drawing of 3D models is a classic artistic technique and remains an important problem in Non-Photorealistic Rendering [1]. A good line drawing can convey the shape geometry without using other cues like shading, color, and texture. Frequently, a few good lines are enough to convey the main geometric features [2]. There are several techniques for depicting shapes with lines, but no single method has proved to be the best for an arbitrary model or viewing position.

Object contours or *silhouettes* are perhaps the most basic type of view-dependent line. Although they may not capture all relevant geometric features in an object, any line drawing should contain the visible boundaries of the object [4]. *Ridges and valleys* [5] are found by computing principal curvatures and principal directions, and their derivatives. They are second-order curves that complement contour information. Since their definition only takes into account the geometry of the model, ridges and valleys are view-independent features. This can cause animation artifacts as these features appear to be rigid, independent of the viewing point. *Suggestive contours* [6] are view-dependent lines that naturally extend contours at the joints. They also depend on second-order information, being based on the radial curvature in the view direction and its derivative. However, suggestive contours do not appear at elliptic regions (where the Gaussian curvature is positive). Thus, convex features cannot be depicted with suggestive contours. Moreover, they do not include silhouettes, which must be computed separately. *Apparent ridges* [3] is a recent technique that seems to produce good results. With a single mathematical definition of what a good line is, apparent ridges depict most features that are captured by other definitions and some additional features not captured before, and its lines are view-dependent. Apparent ridges combine both second-order information and view-dependency: they are based on a view-dependent curvature that plays an analogue role for apparent ridges as the curvature does for ridges and valleys. Another advantage is that contours are a special case of apparent ridges and so do not require an additional rendering step.

Unlike the original method [3], which works entirely over the mesh in object space, our method computes the curvature data needed to find apparent ridges in an object-space phase and an image-space phase. This split allows us to use vertex shaders and pixel shaders to compute each part in the GPU, exploiting GPU processing power and parallelism.

To compute apparent ridges, one must first compute the view-dependent curvature q_1 and its derivative $D_{t_1}q_1$ in the maximum view-dependent principal direction t_1. Apparent ridges are the maxima of q_1 in the t_1 direction, that is, the points where $D_{t_1}q_1 = 0$ and $D_{t_1}(D_{t_1}q_1) < 0$. In the original method, all features are computed in object space, directly on the mesh. It is simple to write a vertex shader to compute q_1 and t_1. However, since the estimation of $D_{t_1}q_1$ on the mesh depends on the values of adjacent vertices, it is not possible to access them in a single vertex pass.


original method


our method

In our method, we still compute q_1 and t_1 on the mesh, but we estimate the zeros of $D_{t_1} q_1$ in image space. This is a reasonable choice because t_1 is, by definition, a screen vector. First, q_1 and t_1 are computed in a vertex program and color-coded in the framebuffer (see the middle picture in the banner figure). Then, the zeros of $D_{t_1} q_1$ are estimated on the image using edge detection via a Laplacian-like adaptive filter that considers the t_1 direction. We implemented this step in a fragment shader. We also included a boundary detector in this fragment shader for handling models with boundaries (like the tablecloth).

As shown in the pictures above, our method produces images that are quite similar to the ones produced by the original method: apparent ridge lines appear generally in the same place Slight differences are due to the nature of the estimation. Some features are better captured by our method (see the tablecloth), but some features are not captured in image space when screen resolution is too high (see some gray lines at the brain). While the images produced by both methods are equally pleasant, we believe that ours are a little more expressive due to the pixel-level estimation.

On the tested models, we have seen speedups ranging from 3 to 8 times faster, using a fixed image resolution. We also noticed that the original method is more sensitive to the mesh size, while our method is sensitive to the image size.

As future work, we intend to experiment with other filters to improve image quality and extend this method to other lines, such as suggestive contours.

REFERENCES

[1] B. Gooch and A. Gooch, *Non-Photorealistic Rendering*. A K Peters, 2001.

[2] M. C. Sousa and P. Prusinkiewicz, "A few good lines: suggestive drawing of 3d models," *Computer Graphics Forum*, vol. 22, no. 3, pp. 327–340, 2003.

[3] T. Judd, F. Durand, and E. Adelson, "Apparent ridges for line drawing," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 19, 2007.

[4] A. Hertzmann, "Introduction to 3d non-photorealistic rendering," in *Non-Photorealistic Rendering (SIGGRAPH'99 Course Notes)*, 1999.

[5] K. Na, M. Jung, J. Lee, and C. G. Song, "Redeeming valleys and ridges for line-drawing," in *Advances in Multimedia Information Processing*, Lecture Notes in Computer Science, vol. 3767, pp. 327–338, 2005.

[6] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 848–855, 2003.

This work is part of the first author's M.Sc. work at IMPA. The authors are partially supported by CNPq. This work was done in the Visgraf laboratory at IMPA, which is sponsored by CNPq, FAPERJ, FINEP, and IBM Brasil.