Apparent Ridges for Line Drawing

HARIHARA GOWTHAM JETTI, Institut Polytechnique de Paris, France

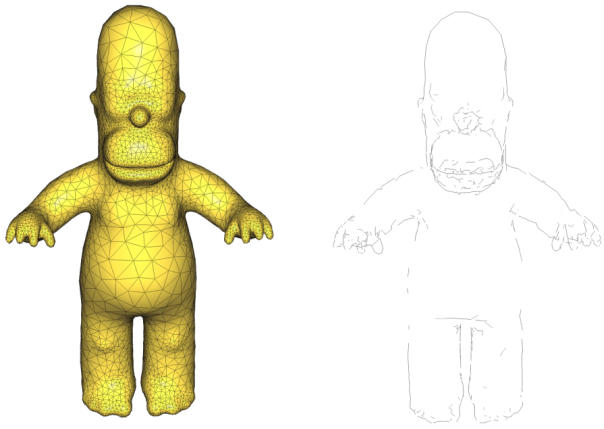

Fig. 1. A mesh along with the line drawing generated using my implementation

1 ABSTRACT

The paper introduces a new approach to defining and drawing feature lines on 3D shapes based on the concept of "apparent ridges". The authors observe that humans perceive shape primarily based on variations in surface normals and the lines that change depending on the viewing direction are effective in conveying smooth surfaces. They define view-dependent curvature as the variation of the surface normal with respect to a viewing screen plane and define the set of points on a surface that maximizes view-dependent curvature as "apparent ridges". The authors propose a formal definition of apparent ridges and develop an algorithm to detect them automatically. The paper demonstrates that apparent ridges encompass or enhance aspects of several other feature lines and can be used to create high-quality line drawings of 3D meshes.

2 INTRODUCTION

Drawing accurate and expressive line drawings of 3D shapes is an important problem in computer graphics, vision, and design. Line drawings can convey essential information about the structure, shape, and texture of an object in a concise and intuitive way and can be used for a variety of applications, including illustration, animation, engineering, and science. However, creating effective line drawings of complex 3D models is challenging, as there are many possible ways to extract and stylize the lines, and no single approach can capture all the desired features and aesthetics. In this report, we focus on work by Judd et al. [1], which introduced a new approach called "apparent ridges for line drawing". We explain the main ideas and methods behind this approach and evaluate its impact and relevance in the field.

The motivation behind the apparent ridges approach is to provide a more comprehensive and perceptually-based definition of feature lines that captures the essence and richness of 3D shapes. Previous methods often rely on simple geometric criteria or ad-hoc stylization rules that do not necessarily reflect the visual salience or meaning of the lines. For example, contour lines may be defined as the boundary between the visible and occluded regions of an object, but they may not always indicate the most informative or interesting parts of the shape. Similarly, depth edges may be defined as the boundary between the front and back surfaces of an object, but they may not always convey the desired sense of depth or shape. By contrast, the apparent ridges approach is based on two key observations about human perception: first, that shape is mainly inferred from the variation of surface normals, and second, that view-dependent lines are more effective in conveying the smoothness and curvature of surfaces. These observations lead to a natural definition of apparent ridges as the set of points on a surface that maximize a view-dependent curvature measure. The apparent ridges approach is thus more consistent with how humans perceive and draw 3D shapes and has the potential to produce more expressive and informative line drawings. In the following sections, we describe the technical details and implementation of the apparent ridges approach and discuss its strengths and limitations in various applications.

3 RELATED WORKS

There is quite a lot of research going on in the area of line drawings based on 3d models. These can be classified into learning-based and non-learning-based. In this report, we will discuss the main and classic works in this area. One of the popular methods is Canny Edge detection [2], which talks about a method for detecting edges in images using first-order derivative operators such as Sobel, Roberts, and Prewitt operators. The paper presents a mathematical formulation for edge detection and defines the concept of edge as the locus of points with high gradient magnitude in the image. Another

related and powerful technique, called Suggestive contours[3], is introduced for conveying shapes -they provide a way to automatically extract and highlight the most salient features of a 3D object's surface, such as ridges, valleys, and corners. This is accomplished by computing a set of 2D curves that lie on the object's surface and emphasize the areas of high curvature. Another work called Sketch-Based 3D Shape Retrieval using Convolutional Neural Networks by Yi et al.[4] proposes a novel approach to 3D shape retrieval using Convolutional Neural Networks (CNNs) and hand-drawn sketches as input - a deep learning architecture that can effectively encode the 3D shape information from 2D sketches and perform accurate retrieval of 3D shapes from a large database. They show that their results outperform state-of-the-art methods for sketch-based 3D shape retrieval on benchmark datasets. DeCarlo et al. [5] discuss about a new technique for real-time rendering of suggestive contours in 3D models. The technique combines temporal coherence with previous algorithms that use view-dependent curves and creases to generate suggestive contours. They propose a new data structure called the temporal coherent edge mesh (TCEM) to store information about the contours, which is then used for efficient rendering. The resulting system enables interactive exploration of 3D models with improved visual fidelity and temporal coherence of the contours. Hertzmann [6] proposed a non-photorealistic rendering technique which aims to create visual output that does not look like a photograph or a realistic image but rather like a hand-drawn or stylized artwork. One important aspect of NPR is the use of silhouettes and outlines to convey the shape and form of 3D objects. By emphasizing the contours of an object and reducing or removing its internal details and textures, silhouettes and outlines can simplify complex scenes and make them easier to understand and interpret. Similarly, a few recent works using learning-based methods can be seen in [8] and [9]. In this way, there is a lot of motivation to develop new techniques for line drawing, and in this report will discuss in depth the Apparent ridges for line drawing.

4 TECHNICAL DETAILS

Various approaches have already been described. In this report, we discuss the apparent ridges for line drawing. This is a non-photo-realistic rendering technique irrespective of illumination and material properties of the image/model based only on normals.

4.1 View-dependent curvature

In order to identify the important lines on the surface of a given mesh, a view-dependent curvature must be defined, which corresponds to significant changes in the surface normals of a 3D mesh when viewed from a particular direction. This is a useful approach as it suppresses details that are not necessary for conveying the overall structure. The concept is based on the observation that human perception is sensitive to changes in surface shading and that such changes are often due to variations in surface normal. By measuring the variation of the surface normal with respect to a viewing direction, the view-dependent curvature can be used to identify important lines on the surface of the mesh.

The various steps involved in finding the view-dependent curvature is shown in Algorithm 4.

Algorithm 1 View dependent curvature computation

1: **for** each vertex v in the mesh **do**
2: **procedure** COMPUTE_VIEW-DEPENDENT_CURVATURE(v, normal, principle curvatures, camera_position)
3: Define view direction from the vertex to the camera position
4: Define the view-dependent curvature operator Q based on the view direction
5: Maximum view-dependent curvature is the square root of maximum eigenvalue of Q.transpose()*Q
6: Maximum view-dependent principal direction is the maximum singular vector of Q

4.2 Estimating its derivative

The view-dependent curvature derivative ($D_{t1}q1$) is a measure of the rate of change of view-dependent curvature in a given direction on a 3D mesh. It is defined as the derivative of the view-dependent curvature with respect to a tangent vector on the surface. The tangent vector is a unit vector that lies on the tangent plane of the surface at a given point and is orthogonal to the surface normal. The view-dependent curvature derivative is useful for identifying rides or valley-like features or edges on the surface, which are areas of high curvature change. It is very useful in detecting and emphasizing the view-dependent changes in surface curvature, the approach can highlight the contours and ridges of the object that are most relevant to its overall shape for a viewing direction.

Different steps involved in computing the derivative of the view-dependent curvature is shown in Algorithm 2.

Algorithm 2 Estimating the view-dependent curvature derivative

1: **for** each vertex v in the mesh **do**
2: **procedure** VIEW-DEPENDENT_CURVATURE_DERIVATIVE(v, view-dependent curvature, view direction, adjacency graph)
3: Take the triangles adjacent to v and define two points on the two edges by linear interpolation in max view-dependent curvature direction
4: Define view-dependent curvature at those two points on the edges
5: Project the curvature to get the derivative of the view-dependent curvature derivative

Discussion of parameters and their effects on line drawing quality

5 METHODOLOGY & IMPLEMENTATION

We begin by computing the normal at each vertex, which is the weighted average of all the adjacent faces or triangles containing the vertex. Next, we calculate the principle curvatures and their respective directions at each vertex. To create the apparent ridges, we need to identify and extract the locus of points on the surface of the 3D mesh where the view-dependent curvature is the absolute maximum. This is achieved by computing the view-dependent curvature and its derivative using the camera position, maximum, and minimum principal curvatures. Subsequently, we find the intersection of points of the gradient of the curvature with the surface.

These intersection points are marked as ridges, which represent the visually sharp features where the surface normal change should be maximum with respect to the viewing direction. Finally, we connect these points with neighbouring points using line segments to create a meaningful overall structure and shape of the 3D model.
The main steps in the implementation are the following:

(1) Pre-processing the input mesh
(2) Computing view-dependent curvature and its derivative
(3) Identifying the apparent ridges
(4) Post-processing

5.1 Pre-processing the input mesh

To accurately generate the apparent ridges, it is necessary to establish a fixed camera position. This can be achieved by computing a bounding volume sphere with the mean of all the vertices as the center, and a radius proportional to the farthest vertex from the center. Subsequently, a ray is shot from the center of the screen, and the first point of the sphere that gets hit by the ray is selected as the camera position. For optimal viewing of the mesh, the radius can be increased or decreased with a key press as desired, and the camera can be placed in a position that provides the required view. Then, to save running time, we pre-compute the normals, principal curvatures and adjacency list of triangles to each vertex.

5.2 Computing View-dependent curvature

The traditional definition of principle curvature is based on identifying the maximum and minimum curvatures and their respective directions at a given point on a surface. Mathematically, these curvatures are represented by the eigenvalues of the shape operator S and their directions by the eigenvectors. By definition, the maximum principle curvature corresponds to the maximum amount of surface normal change at a point, while the minimum principle curvature corresponds to the minimum change. These extremal curvatures and directions are often used to identify ridges on a surface. In order to define a view-dependent curvature, a specific viewing direction is introduced, and the change in surface normal is measured when projected onto the screen as shown in 2. Since objects are foreshortened when viewed from a particular angle, even small displacements on the screen can correspond to larger normal changes on the object. This effect is particularly noticeable at locations where the object turns away from the viewpoint. Thus, the view-dependent curvature Q is computed by multiplying the surface normal with the inverse projection matrix.

$$Q = SP^{-1}$$

And here is the step-by-step algorithm 4 to compute the view dependent curvature

5.3 Computing the derivative of view-dependent curvature

The computation of the view-dependent curvature derivative involves estimating $D_{t1}q1$ using finite differences, as outlined in the paper. At each vertex of the mesh, we compute the view-dependent curvature at two points, w and w', located on the edges of its adjacent

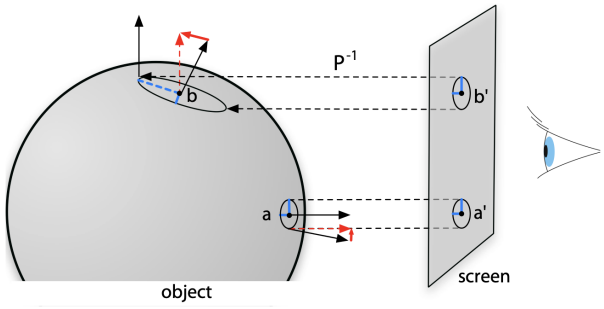

Fig. 2. Taken from the paper, shows projected normal change

triangles in the absolute maximum principle direction. The finite difference between the vertex and the two w points is then averaged to obtain the derivative. To find the view-dependent curvature at point w on an edge, we use linear interpolation between the triangle's two vertices. Next, we compute the projected view-dependent curvature derivative by subtracting the interpolated view-dependent curvature from the current view-dependent curvature at vertex p and dividing this difference by the projected distance between p and w along the principle direction. This process is repeated for all adjacent triangles to vertex p, and the resulting estimates are averaged to obtain the final estimate of the view-dependent curvature derivative. It is sufficient to compute only two estimates, which we obtain before terminating the algorithm. The overall step-by-step algorithm used to compute the derivative of the view-dependant curvature is shown in Algorithm 2.

5.4 Identify and draw the apparent ridges

During the identification of apparent ridges, we first ensure that the edges or vertices being examined do not belong to the faces that are behind and not visible from the camera position. We then follow the approach presented by Ohtake et al.[] and locate the zero crossings of the previously computed view-dependent curvature derivatives on the mesh. This involves checking the view-dependent curvature directions at the endpoints (vertices) of each edge of a face. If the directions point in the same direction, there is no zero crossing, and the edge can be directly marked as an apparent ridge. However, if the directions point in different directions, indicating an angle greater than 90 degrees, then a zero crossing exists. In such cases, we interpolate the edges and determine the location of the zero crossing using the derivative values at each vertex. And finally, we mark these edges as apparent ridges.
And here is the step-by-step algorithm 3 to compute the derivative of the view dependent curvature
The main algorithm is as follows:

6 RESULTS

6.1 Observations

- This method captures the variations in surface normals that are crucial for shape perception and the view-dependent lines (ridges & valleys) that are more informative for smooth surfaces.

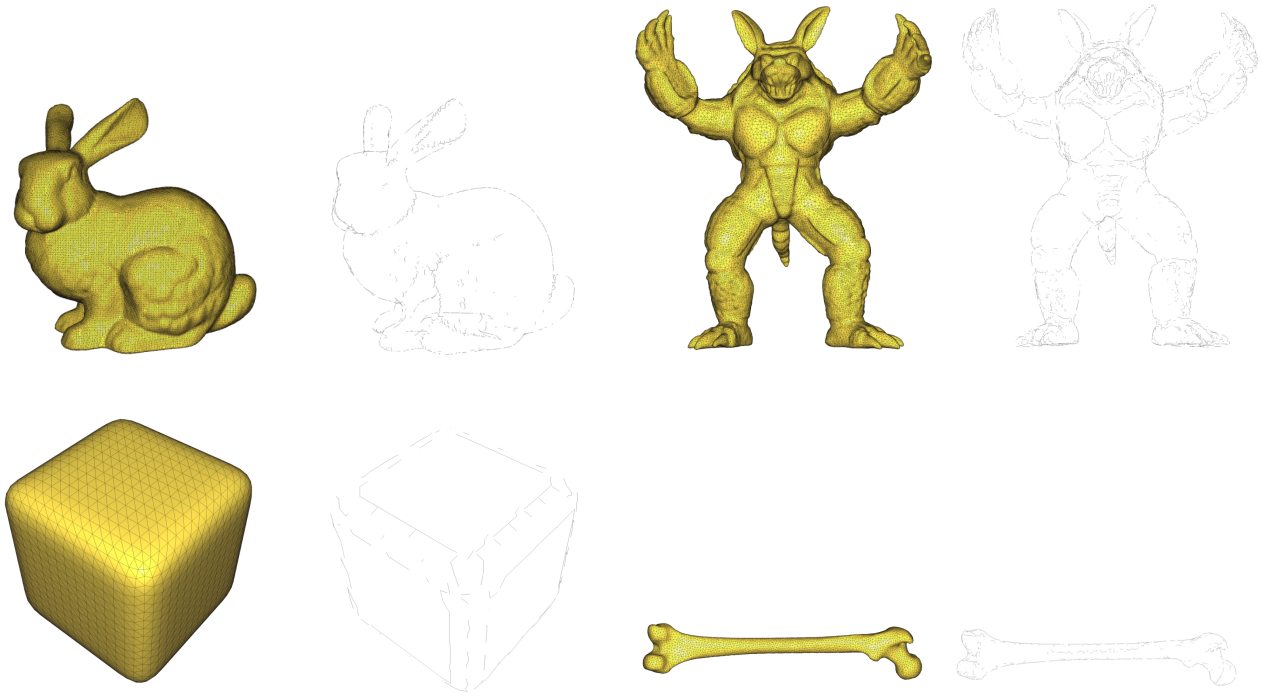

Fig. 3. Various results generated using my implementation (Left: the mesh, Right: the extracted line drawings)

Algorithm 3 Identifying points as apparent ridges

```
for each face f in the mesh do
    procedure FIND_APPARENT_RIDGES(f, view-dependent curvature, view direction, view-dependent curvature derivative, threshold)
        if f is not a back face then
            if View-dependent curvature is not below threshold then
                if View-dependent curvature directions of the two vertices of any edge of f is less than 90 degrees then
                    Interpolate the edges to find the ridges or valleys as endpoints
                    Save the vertex and the endpoint as edges
```

Algorithm 4 Apparent Ridges for Line Drawing

```
procedure APPARENT_RIDGES(Mesh m, camera_position, adjacent graph, threshold)
    for each vertex v in m do
        Calculate normal, max & min principle curvatures and directions at v
        procedure COMPUTE_VIEW-DEPENDENT_CURVATURE(v, normal, principle curvatures, camera_position)
        procedure VIEW-DEPENDENT_CURVATURE_DERIVATIVE(v, view-dependent curvature, view direction, adjacency graph)
        procedure FIND_APPARENT_RIDGES(f, view-dependent curvature, view direction, view-dependent curvature derivative, threshold)
        Draw all the edges marked as apparent ridges
    procedure POST-PROCESSING(Edges)
```

- Although the apparent ridges approach is effective for generating line drawings of 3D shapes, it has some limitations that need to be addressed. These include sensitivity to the quality of the input mesh, which can affect the accuracy and detail of the resulting line drawing. Additionally, careful parameter tuning and post-processing may be necessary to achieve the desired effect.
- The method may not capture all of the feature lines that are important for specific tasks or aesthetic preferences. These challenges may require the use of complementary methods or customization for particular applications.
- It can be difficult to determine an optimal value for the threshold parameter used to control how much simplification should occur without affecting the accuracy of the output.

7 CONCLUSION AND FUTURE DIRECTIONS

In this project, we have implemented an technique for rendering apparent ridges (& valleys) for line drawings. It is an effective method to generate visually appealing and informative line drawings of 3D shapes based on the current view. However, to achieve the desired effect, some challenges and trade-offs need to be considered.

One such challenge is that the technique may be sensitive to the sampling density, resolution, and noise in the input mesh. Therefore, careful parameter tuning and post-processing may be required. Additionally, while apparent ridges capture variations in surface normals that are crucial for shape perception, they may not always capture all relevant feature lines for certain tasks or aesthetics. As a result, combining this technique with other methods or customizing it for specific domains may be necessary. Despite these limitations, implementing an apparent ridges approach represents a significant contribution to line drawing generation and opens up new possibilities for visual communication and design of 3D shapes. And also apparent ridges approach has been shown to be effective in a range of applications, including artistic rendering, technical illustration, animation, and scientific visualization, and can produce visually appealing and informative line drawings that convey the essential features of the shapes.

This whole project is implemented in C++ using the libIGL [7] library by Alec Jacobson et. al and tested on an 8-core MacBook M1 Pro with 16GB memory. Almost all the results took around 2-3 seconds to generate.

REFERENCES

[1] Judd, Tilke, Frédo Durand, and Edward Adelson. "Apparent ridges for line drawing." ACM transactions on graphics (TOG) 26, no. 3 (2007): 19-es.

[2] Canny, J., 1986. A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6), pp.679-698

[3] DeCarlo, Doug, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. "Suggestive contours for conveying shape." In ACM SIGGRAPH 2003 Papers, pp. 848-855. 2003.

[4] Wang, Fang, Le Kang, and Yi Li. "Sketch-based 3d shape retrieval using convolutional neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1875-1883. 2015.

[5] DeCarlo, Doug, Adam Finkelstein, and Szymon Rusinkiewicz. "Interactive rendering of suggestive contours with temporal coherence." In Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, pp. 15-145. 2004.

[6] Hertzmann, Aaron. "Introduction to 3d non-photorealistic rendering: Silhouettes and outlines." Non-Photorealistic Rendering. SIGGRAPH 99, no. 1 (1999).

[7] Jacobson, Alec, and Daniele Panozzo. "Libigl: Prototyping geometry processing research in c++." In SIGGRAPH Asia 2017 courses, pp. 1-172. 2017.

[8] Liu, Difan, Matthew Fisher, Aaron Hertzmann, and Evangelos Kalogerakis. "Neural strokes: Stylized line drawing of 3d shapes." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14204-14213. 2021.

[9] Liu, Difan, Mohamed Nabail, Aaron Hertzmann, and Evangelos Kalogerakis. "Neural contours: Learning to draw lines from 3d shapes." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5428-5436. 2020.