Stereo-Consistent Apparent Ridges

A Gradient-Based Geodesic Search for View-Dependent Lines in Virtual Reality

Tim Tian

Delft University of Technology

TU Delft

Stereo-Consistent Apparent Ridges

A Gradient-Based Geodesic Search for View-Dependent Lines in Virtual Reality

---

THESIS

submitted in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Tim Tian

Advisor: Prof. Dr. Elmar Eisemann
Supervisors: Dr. Petr Kellnhofer & Amir Zaidi
Faculty: Electrical Engineering, Mathematics and Computer Science (EEMCS)

Abstract

Virtual Reality (VR) demands high-performance stereoscopic rendering to maintain immersion, yet current techniques for Non-Photorealistic Rendering (NPR) often struggle with stereo-consistency. Despite line drawings, specifically apparent ridges, being a powerful tool for conveying shape and depth with minimal visual clutter, their view-dependent nature present a significant challenge in stereoscopic settings. As the viewpoint changes, view-dependent lines can have discrepancies between left and right viewpoints, triggering binocular rivalry and undermining the user's depth perception.

This thesis proposes a framework for synthesizing stereo-consistent lines through a gradient-based geodesic search in world-space. Rather than attempting to match existing view-dependent lines in stereo-viewpoints, we introduce a three view correspondence algorithm. Using the midpoint of the stereo viewpoints as an anchor, we use an iterative pathfinding algorithm that performs geodesic walks along the surface gradient of the curvature field. This process traces the transition of apparent ridges, a view-dependent line, from the center reference point to the stereo viewpoints, identifying consistent surface regions across the mesh.

To enhance depth and shape perception for the user, we extend this algorithm to apply hatching lines on the marked surface regions. By rendering curvature-aligned hatching lines along the geodesic paths, we provide additional surface cues that apparent ridges cannot convey. This method ensures that both eyes are presented with the same lines and information, eliminating most cases of binocular rivalry.

The efficacy of this approach is measured using a user study conducted in VR, alongside a performance analysis to ensure the algorithm meets the 90 FPS requirement of VR. Our results demonstrate that there is a statistically significant improvement in 3D shape and depth perception for users, while maintaining the visual comfort that apparent ridges provide.

Preface

First, I would like to thank my parents, Biao and Weiping, for raising and supporting me throughout tough moments and happy times. Life has taught me that nothing is certain and sometimes what we wish for will not work out, but I hope that despite these uncertain moments I will be able to repay you a life full of certainty and happiness. Second, I would like to thank Wakana, her support throughout this thesis has helped me grounded and encouraged throughout the long hours of writing and research. Third, I want to thank Dr. Petr Kellnhofer and Amir Zaidi for their guidance and feedback throughout the whole project, whom without I would not have completed this journey with and I am truly thankful for everything you have given me. I would also like to thank Prof. Dr. Elmar Eisemann for their interest and willingness to help find a suitable topic. Additionally, I am thankful for Dr. C.A Raman for appearing on my thesis committee despite the scheduling changes. Finally, I would like to thank all my friends and family who have been with me throughout the years.

Tim Tian
April 2026

Contents

List of Figures

List of Tables

1

Introduction

Virtual Reality (VR) technology has gained much traction in recent years, especially in fields like entertainment, education, medical training, architectural visualisation, and industrial design. At its core, VR aims to create an immersive experience for users in synthetic 3D environments. Unlike traditional displays, VR systems present stereoscopic images directly to each eye through head-mounted displays, generating the perception of depth within virtual spaces. This immersive quality of VR makes it valuable for applications where spatial understanding is crucial, or an immersive experience is wanted.

However, presenting convincing immersion in VR has shown to be technically difficult. The human visual system, while great at seeing depth and shape, is sensitive when it comes to inconsistencies between the images of two eyes, making VR applications more demanding of systems in ways that traditional graphics applications are not. To maintain an immersive experience for users, VR applications should render high-quality images at consistent frame rates exceeding 90 frames per second, with minimal latency between movement and display [27]. These performance requirements are aggravated by the need to render each frame twice, once for each eye, doubling the rendering computation. Furthermore, visual discrepancies between the images presented to each eye can trigger binocular rivalry, a phenomenon where the brain is unable to fuse the two images presented into a 3D percept. Binocular rivalry often manifests as double vision, where the images are rapidly alternating creating a flickering effect. This leads to visual discomfort, breaking immersion and potentially causing eye strain, nausea, or headaches. Thus, the combination of high performance demands and perceptual constraints makes graphics algorithms for VR a challenging problem.

While much of graphics research has been focused on photorealistic rendering that simulates physical behaviour, Non-Photorealistic Rendering (NPR) attempts an alternative visual style that does not follow that exact realistic behaviour. Among NPR techniques, line drawings have proven to be a powerful tool for shape and depth cues, whilst offering a minimal way of conveying complex geometry.

Numerous algorithms have been developed to extract meaningful lines from 3D geometry. Lines such as silhouettes mark the boundaries where objects occlude the background, while creases highlight sharp geometric features such as an edge of a cube. Over time, more sophisticated techniques like suggestive contours and apparent ridges extend these concepts to create meaningful lines that enhance depth and shape perception even when they do not correspond to geometric discontinuities. The usage of line drawing in 3D visualizations offer several advantages to photorealisitic rendering: it reduces visual clutter, emphasizes important geometric features, and can run efficiently since line rendering requires fewer computational resources than physically correct photorealistic rendering.

For VR applications, line drawings present an interesting opportunity. The reduced complexity of line drawing techniques could help meet the required performance needed for an immersive VR experience, whilst still conveying depth and shape information. Certain line styles can also enhance VR experience in certain fields, where visual clarity is preferred over realism.

Despite the potential of line drawings in a stereoscopic environment, new challenges arise that do not

exist in single-view rendering. While many line drawing techniques are view-independent, meaning the position of the lines on the surface are fixed, most sophisticated lines, such as apparent ridges, are view-dependent, where the position of the lines on the surface change depending on the observer's viewpoint. In a stereoscopic environment, the offset between left and right viewpoints can cause these lines to appear at different locations on the surface of the object for each eye. This lack of stereo consistency between views results in lines that convey misleading depth cues and can trigger binocular rivalry. Furthermore, even view-independent lines can exhibit inconsistencies. Despite the fixed loci of lines for view-independent lines, different occlusion for left and right viewpoints, where a line is visible in one eye but occluded in the other, can still produce inconsistent imagery between eyes.

Existing approaches to stereo-consistent line drawing focus on matching view-dependent lines that have already been computed independently for each eye. These methods operate in either screen-space or use proximity-based world space matching to identify which line in one view corresponds to which line in the other view, then filter inconsistent matches. However, these matching techniques have fundamental limitations. When view-dependent features shift substantially across the surface between viewpoints, establishing reliable correspondence becomes difficult, and the resulting matched lines may appear fragmented or inconsistent between eyes.

1.1. Research Questions

Based on the challenges identified regarding stereo-consistency and visual comfort in Virtual Reality, this thesis aims to answer the following research question:

How can the synthesis of stereo-consistent lines inspired by line drawing techniques enhance the 3D depth and shape perception of users in Virtual Reality?

To address this research question, we first create a matching algorithm capable of matching one such line drawing technique, apparent ridges, from stereo views. From this algorithm, we derive stylized hatching lines that provide additional shape and depth cues. We further optimize the algorithm to be efficient and suitable for VR. Then we determine the performance of these new hatching lines by conducting a user study in a VR environment. To structure the progress of this thesis, we further divide the main research question into the following sub-questions:

1. How can hatching lines be created from matched apparent ridges in a stereo context?
2. What methods can be used to ensure stereo-consistency within the performance constraints of Virtual reality?
3. To what extent do these new hatching lines improve a user's ability to perceive depth and shape in Virtual Reality?

To answer these questions, we propose a new line drawing technique that is naturally stereo-consistent, originating from apparent ridges. Instead of calculating and rendering view-dependent lines independently for each eye, we propose a center-view synthesis approach. By using the midpoint of stereo viewpoints, we establish a non-biased reference set of apparent ridges. To ensure these new lines remain stable and consistent across both eyes, we introduce an iterative pathfinding algorithm in world-space. This algorithm matches apparent ridges from the central reference apparent ridges to the stereo-viewpoints, tracing continuous paths on the surface between them. We then render hatching lines along these paths, ensuring that both eyes are presented with lines that correspond to the same 3D location on the surface. This approach effectively makes the hatching lines view-independent and less prone to binocular rivalry.

Furthermore, we utilize hatching lines to render along the path of our pathfinding algorithm. Hatching lines have the benefit of providing directional information of the surface orientation, further enhancing the shape and depth cues in line drawings. By drawing hatching lines along the path found in the pathfinding algorithm, we provide additional depth and shape cues over surface geometry that can be used in a stereoscopic setting. Unlike traditional apparent ridges, which are often thin and inconsistent in stereo environments, these synthesized hatching lines are naturally stereo-consistent, provide additional depth and shape clarity in VR.

1.2. Contributions

The primary contributions of this work are:

- A mathematical formulation for finding the closest apparent ridge on any point of an object.
- A three view correspondence algorithm in world-space, capable of iteratively matching view-dependent lines using a gradient guided geodesic walk and extending the path beyond them.
- A novel application of hatching lines as a visualization tool for surface paths generated from our three view pathfinding algorithm, that integrate apparent ridge differences between stereo viewpoints.
- A user study evaluating the efficacy of these lines against traditional apparent ridges.

2

Background and Related Work

This chapter establishes the theoretical and technical foundations required to perform gradient based geodesic walks on the surface manifold and synthesize stereo-consistent hatching lines in immersive stereo environments. To bridge the gap between abstract 3D geometry and the human visual system, we first examine the mathematical definitions of curvature and epipolar geometry. We then explore the evolution of Non-Photorealistic Rendering, categorizing feature lines by view-dependency and their method of conveying shape. Finally, we analyse the perceptual constraints of the human visual system and the performance requirements of Virtual Reality, providing critical context for why traditional line-drawing techniques often fail to maintain the stereo coherence for a comfortable immersive user experience.

2.1. Basic background

2.1.1. Curvature

Curvature is a fundamental concept in differential geometry that quantifies how much a surface bends in space. Curvature quantifies the rate at which the unit tangent vector changes as you move along the curve, with larger or higher curvature meaning a sharper bend. At any point $p \in \mathbb{R}^3$ on a surface S, we define the tangent plane T_pS and a unit surface normal $n(p)$. Surface curvature is characterized by how the normal $n(p)$ changes as we move in a specific direction u on the tangent plane. To compute curvature, we use two quadratic forms that describe local geometry: the First Fundamental Form and the Second Fundamental Form [34]. The First Fundamental Form I represents the inner product on the tangent space. For two tangent vectors $u, v \in T_pS$:

$$\mathrm{I}(u, u) = u \cdot v \tag{2.1}$$

It measures the distances and angles on the surface. When $u = v$, the form $\mathrm{I}(u, u) = ||u||^2$ provides the squared length of the vector. The Second Fundamental form II measures the shape or curvature of the surface. The Second Fundamental Form II measures how the surface curves away from its tangent plane by quantifying the rate of change of the normal vector. For a tangent direction u, we measure how the normal n changes as we move along the surface in that direction. Mathematically:

$$\mathrm{II}(u, u) = -\frac{dn}{du} \cdot u \tag{2.2}$$

where $\frac{dn}{du}$ represents the directional derivative of the unit normal vector n in direction u. If the normal changes rapidly as we move in direction u, the Second Fundamental Form, and thus the curvature, is large. The normal curvature $\kappa_n(u)$ is the curvature on a point p on the surface with tangent vector u. It is defined as the ratio of the Second Fundamental Form to the First Fundamental Form. For a unit tangent vector u (i.e. $||u|| = 1$), it is defined as

$$\kappa_n(u) = \frac{\mathrm{II}(u,u)}{\mathrm{I}(u,u)} = \mathrm{II}(u,u)$$

As we rotate the direction u on the tangent plane, $\kappa_n(u)$ changes. The maximum and minimum values of this function are called the principal curvatures, noted as $\kappa_1(p)$ and $\kappa_2(p)$. The direction in which these extrema occur are the principal directions, $e_1(p)$ and $e_2(p)$. These directions are always orthogonal to each other in the tangent plane.

The principal curvatures provide information on the local shape of the surface. Gaussian curvature, the product of the two principal curvatures $K(p) = \kappa_1(p)\kappa_2(p)$ can indicate a saddle like surface when $K(p)$ is negative, or an elliptic point when $K(p)$ is positive. An illustration detailing a saddle surface can be found in Figure 2.1.

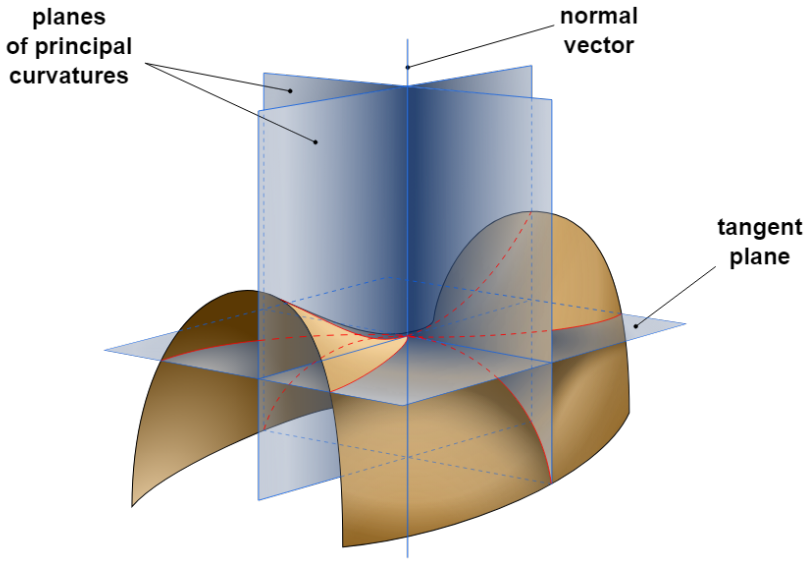


Figure 2.1: An example saddle surface, showing the surface normal, tangent plane and principal direction planes. Source: Wikimedia, by: Eric Gaba, License: CC-BY-SA 3.0.

2.2. Line drawing

Line drawing in the context of Non-Photorealistic Rendering (NPR) aims to simplify complex scenes into sparse, meaningful geometric representations. By using feature lines, line drawings can communicate shape, depth, volume, and spatial relationships more efficiently in certain applications [34, 9, 10]. Feature lines in this domain are generally classified in two categories, based on their relation with the observer: view independent lines and view dependent lines.

2.2.1. View independent lines

View independent lines are static regardless of the viewing position of the user and viewing direction. This results in the same lines being drawn when viewed from different angles. Some examples of view independent lines are the following:

Creases

Creases are surface lines that occur where the surface normal changes discontinuously or where the change in normal is abrupt [8, 34]. These lines represent sharp edges or folds on a surface, representing features such as edges or sharp bends on smooth surfaces.

Ridges and Valleys

Ridges and valleys are feature lines that are defined based on the surface's principal curvatures and their directions. A ridge occurs along the points where the principal curvature contains a local maximum

along its corresponding principal direction, whereas a valley occurs where the principal curvature has a local minimum [16, 34, 28]:

- Ridges: Points where the largest principal curvature κ_1 has a local maximum along its direction.
- Valleys: Points where the smallest principal curvature κ_2 has a local minimum along its direction.

2.2.2. View dependent lines

View dependent lines are lines that exist and change dependent on the viewing position and direction, unlike view-independent lines. This results in different lines being drawn when an object is viewed from different angles. Some view dependent lines are explained below:

Silhouettes

Silhouettes, or specifically exterior silhouettes, are the lines on the boundary of the projected object shape and the background. Additionally, occluding contours, or interior silhouettes, are the lines where any depth discontinuities are present, not only against the background, [34, 16]. These lines can mathematically be defined as points where the view direction v is perpendicular to the normal of the surface n. Thus, where $n \cdot \mathrm{v} = 0$. An example of both exterior and interior silhouettes can be found in Figure 2.2.

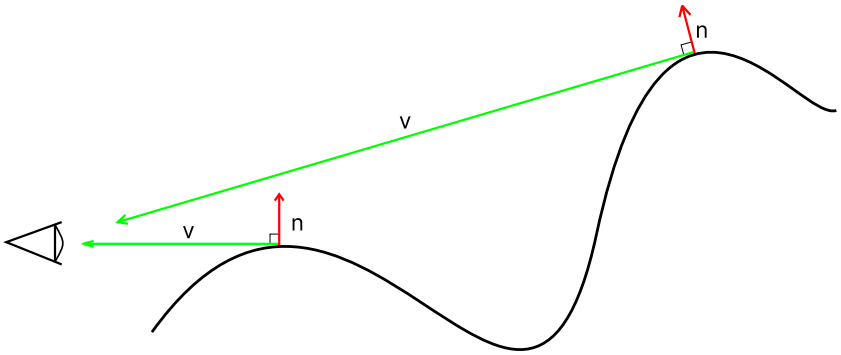


Figure 2.2: Example of occluding contours and silhouettes.

Suggestive Contours

Suggestive contours are informally known as 'almost' contours. These are points where a contour could be found in nearby viewpoints, providing more information on the shape of the object. There are three definitions of suggestive contours: contours in nearby viewpoints, minima of $n \cdot v$, and zeros of radial curvature [13, 11]. Suggestive contours have also been used to generate other families of lines such as suggest highlights and principal highlights, complementing suggestive contours [12].

Apparent Ridges

Apparent ridges are view-dependent lines that emphasize regions where the surface normal n changes most rapidly from the perspective of the viewer. Unlike traditional ridges, apparent ridges take into account the viewer's position v, making them more consistent with visual perception of shape.

To understand apparent ridges, we first need to consider how surface curvature appears from specific viewpoints. At any point p on the surface, we can measure how fast the surface normal $n(p)$ changes as we move across the surface. However, for apparent ridges, we do not consider the 3D normal, but instead project the 3D normal onto the viewing plane to get a 2D view-dependent normal. The rate of change of this projected normal as move along the surface gives us view-dependent curvature.

Formally, this is captured by a view-dependent second fundamental form A_v, which generalizes the second fundamental form (Equation 2.2, Section 2.1.1) to account for the position of the viewer. While the classical second fundamental form measures how the 3D normal changes in tangent directions, the view-dependent second fundamental form measures the change in **projected** normal as we move along the surface in different tangent directions. Just as the classical second fundamental form has principal curvatures and directions, A_v has principal view-dependent curvatures $q_1(p)$ and $q_2(p)$ (with

$q_1(p) \geq q_2(p)$) and corresponding principal directions $t_1(p)$ and $t_2(p)$ that form an orthonormal basis in the tangent space T_p.

This thesis follows the mathematical definition of apparent ridges by Judd, Durand, and Adelson [21]. Thus, apparent ridges are defined as the loci of points where the maximum view-dependent curvature $q_1(p)$ achieves a local maximum along its corresponding principal direction $t_1(p)$. Mathematically, this occurs for a point p when the directional derivative of $q_1(p)$ in the direction $t_1(p)$ is zero:

$$D_{t_1} q_1(p) = 0 \tag{2.3}$$

where $D_{t_1} q_1(p)$ denotes the directional derivative of the view-dependent curvature field q_1 along the tangent direction $t_1(p)$ on the surface. While $q_1(p)$ is derived from the projected screen-space normal, it is important to note that the calculation of the directional derivative $D_{t_1} q_1(p)$ is performed in the tangent space of the 3D surface.

2.2.3. Hatching Lines

Hatching lines are a dense set of curves used to convey surface shape, orientation, and texture through varying line density and direction. Hatching lines in NPR have been used to help artists, or non-artists, generate imagerey in different styles [31]. Hatching lines have been used as both view-dependent and view-independent lines. In early real-time methods, hatching lines have been built upon other lines, such as contours [29], suggestive contours [38], or are generated view-independently where cross-hatching lines, a method of hatching, are generated from the surface normal and light direction. Hatching lines have often been aligned with principal curvature directions (κ_1 and κ_2), ensuring that the strokes follow the underlying geometric structure of the surface regardless of viewpoint [31, 19].

2.2.4. Epipolar geometry

Stereo vision relies on the fact that our two eyes observe the world from slightly different perspectives. Epipolar geometry provides the mathematical framework for describing the relationship between these two views and shared 3D points, thus stereoscopic 3D [41, 3, 5, 14].

Consider a setup of two cameras or eyes, O_L and O_R, separated by a horizontal distance known as the baseline. In a standard VR configuration, these cameras are oriented parallel to one another, looking straight ahead. This is distinct from other toe-in configurations, where the two cameras are angled inward. When both cameras observe a single 3D point P, that point projects onto specific locations on each image plane (p_L and p_R). The horizontal difference between two points is known as disparity, which is dependent on the distance between the cameras and the distance to point P. An example of this can be found in Figure 2.3, where the cameras are located at O_L and O_R. The line between the cameras is the baseline. The epipolar plane is the triangle plane formed between the two cameras and the point P and the intersection between the epipolar plane and the two image planes is the epipolar line .

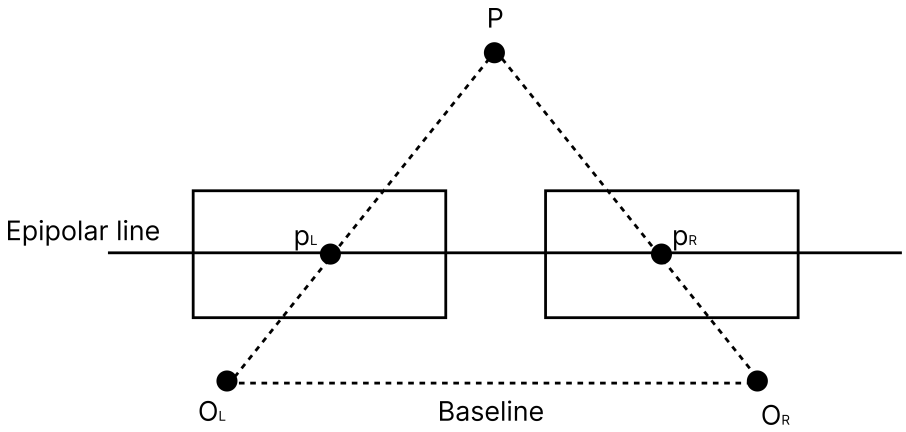


Figure 2.3: Basic setup of epipolar geometry.

2.3. Stereoscopy

Stereoscopy is a technique which creates depth in images, using two different camera views representing each eye. By using the slight differences in each image, more commonly known as binocular disparity, the human brain can infer depth. However, this is not the only way the brain infers depth, occlusion, perspective, and relative size, are examples of monocular cues, where only one eye is needed for depth estimation [3, 30].

2.3.1. Random Dot Stereograms

The most significant evidence for binocular disparity is the Random Dot Stereogram (RDS), introduced by Béla Julesz around 1960. An RDS consists of a pair of images containing identical random noise, where a region of dots in one image is shifted horizontally [22]. When viewed stereoscopically, the brain recognizes the shifted noise patterns and is able to form a 3D shape that 'floats' above all other noise. On the other hand, this test is capable of testing a persons ability to perceive depth through binocular disparity. This inability of perceiving depth through binocular disparity is known as stereo-blindness and a partial inability is known as stereo-deficiency [32, 30]. Estimates suggest that roughly 7% of the population under 60 years old is stereo-deficient or stereo-blind [7]. In the context of stereo imagery, this demographic cannot see the effects of the RDS test, or depth and shape cues from binocular disparity alone. This is why we test our subjects using an RDS test as a measure to ensure that all participants possess a sufficient level of stereopsis. Since this research evaluates the efficacy of line drawing techniques based on their ability to convey depth and minimize binocular rivalry, it is critical to confirm that participants are capable of perceiving the binocular cues provided.

2.3.2. Monocular depth cues

Monocular cues are essential for depth perception, especially at medium and long distances, where binocular cues are less effective. They are especially helpful for gauging relative depth, providing vital information within ones view. Key monocular cues include:

- Occlusion: When one object partially blocks the view of another, the brain interprets the blocking object as being closer.
- Linear perspective: Parallel lines appear to converge as they extend toward the horizon, meeting at a vanishing point, thereby creating a sense of depth.
- Size: Objects appear smaller as they recede into the distance. The brain uses its knowledge of the average size of familiar objects to estimate their relative depth.
- Texture gradient: Fine details and textures on a surface are more distinguishable at close range, becoming less apparent as the object moves further away, reinforcing depth perception.
- Motion parallax: When in motion, closer objects appear to move faster than distant ones, a cue that helps gauge their relative distances.

These monocular cues provide the brain with sufficient information to infer depth in a scene, even when viewed with one eye or at distances where binocular cues are less reliable. However, while monocular cues are powerful, they often work best in combination with binocular cues to achieve more accurate and immersive depth perception.

2.3.3. Binocular depth cues

Binocular cues rely on the brain's ability to combine and infer depth from both our eyes. They provide vital information for accurately estimating depth, as well as enhancing our ability to estimate depth in complex scenes [3, 30]. Mainly two binocular cues are used for estimating depth:

- Binocular disparity: Each eye perceives the world at a slightly different angle and position, our brain uses the two views to combine and gauge the depths of objects in our view.
- Vergence: The degree that our eyes need to turn to view an object. Closer objects require more convergence, and further objects require little or almost no convergence.

The combination of monocular and binocular depth cues is vital to how the human visual system infers depth in the world. Monocular cues provide context to our scene, while binocular cues ensures

precision. The integration of both allows for a robust and comprehensive perception of depth, allowing us to understand complex scenes with accuracy. Together, these depth cues compensate for the weaknesses and limitations of the other, creating an immersive visual experience.

Binocular rivalry

Binocular rivalry is a natural phenomenon that occurs when the brain is presented with two different images, one for each eye, at the same time. Instead of fusing the two images into a single three dimensional view (stereopsis), the brain is unable to merge the conflicting images and instead enters a state of competition where each image suppresses the other to be perceived. Binocular rivalry causes the human visual system to randomly alternate between the two images, resulting in viewing discomfort and an impairment of depth perception [2, 40] This failure of fusion is often caused by the violation of the epipolar constraint. Human stereo-matching often assumes that corresponding features lie along the horizontal epipolar lines [17]. By restricting the search space to these lines, the brain can more efficiently calculate disparity. While the visual system is capable of handling a small degree of vertical disparity up to 45 arcmin, discrepancies beyond this threshold or other significant differences can still trigger binocular rivalry [39].

2.4. Stereo-consistency and stereo-coherence

Stereo-consistency refers to the geometric constraint that a feature drawn in one eye must have a corresponding feature in the other eye that lies on the same epipolar line, this effectively constrains each feature drawn to be the same 3D point in space [17]. **Stereo-coherence** refers to the consistency of visual features perceived in stereoscopic 3D imagery when viewed by both eyes. Stereo coherence ensures that visual artifacts such as lines, contours, and other geometric features appear seamless and aligned when projected onto the left and right eye views. Without coherence, features can cause discomfort or distortions and creates an inconsistent visual experience [23]. It is important to note that stereo-consistency is not the same as stereo-coherence. For example, a line might be mathematically consistent, but become incoherent if it is occluded in only one of the eyes. Conversely, a lack of stereo-consistency almost always results in a lack of stereo-coherence. Thus, binocular rivalry, stereo-consistency, and stereo coherence go hand in hand. Stereo coherence is particularly important for applications requiring stereoscopic displays, where binocular rivalry can lead to visual discomfort, depth misinterpretation, or disorientation.

Achieving stereo coherence in line drawing, where silhouettes, contours, or shading lines must align correctly in 3D space across both views, has been seen as challenging by many. As line drawings are sparse images for conveying the shape of objects, even small discrepancies between the two views can cause misinterpretations of the depth and shape.

2.5. Temporal consistency and temporal coherence

Temporal coherence is the relation and similarity between adjacent frames in rendering. As noted by Akenine-Moller, Haines, and Hoffman [1], in most rendering scenarios the camera and objects move incrementally between time steps. This results in data redundancy, which can be exploited to compute new frames from previous frames [35].

While temporal coherence describes the similarity in data, **temporal consistency** describes the stability of visual output over frames. Temporal consistency is achieved when an algorithm produces smooth, continuous transformation of features between frames [26]. In the absence of temporal consistency, frequent changes between frames can cause flickering and lead to temporal aliasing.

2.6. Stereo-consistent line drawing

Several authors have researched how view-dependent lines can be rendered in a stereo-consistent manner. Naive approaches to line drawings in stereoscopic 3D often lack accuracy or are not stereo coherent. As described by Kim et al. [23] two naive approaches fail when considering stereo coherent suggestive contours. First, the each-eye approach, in which the lines are drawn separately for each eye. This approach causes binocular rivalry to occur due to the view-dependent nature of suggestive contours. Naturally, this is not wanted when considering an immersive VR experience. Alternatively,

the center-eye approach, lacks accuracy, as it draws view-dependent lines from the center of both eyes, rather than considering the view from both eyes. Whilst this method is stereo-consistent and attempts to minimize binocular rivalry as the lines are now stable across viewpoints, rivalry can still occur in similar situations as view-independent lines such as occlusion for one eye [15]. In general, the same issues arise when considering these two naive approaches to other, view independent types of lines.

Kim et al. [23] proposed a novel algorithm for stereo-consistent rendering of suggestive contours, utilizing the epipolar plane and samples of nearby viewpoints to discard stereo-inconsistent lines. By sliding along the epipolar directions in image space, they ensure stereo consistency. However, this method is unsuitable for real-time rendering, as their unoptimized system had only achieved 3 FPS due to the computational overhead of epipolar search and viewpoint sampling.

Another method created by Bukenberger, Schwarz, and Lensch [6] uses an object-space approach for stereo-consistency. They use a precomputed multi-view window and nearest-neighbors for matching occluding contours. By working in object-space, their method maintains topological continuity that screen-space methods lose. Despite this, they only achieved 24 FPS and the reliance on the precomputed multi-view window makes it not a viable solution for VR systems where users can move freely in space. Furthermore, their approach is optimized for boundary features rather than interior surface features like apparent ridges which can appear and disappear.

Lastly, the most recent solution for view-dependent line drawing was developed by He, Wang, and Bao [18]. They built upon the method described by Kim et al. [23] and extend it to not require sampling of nearby viewpoints, but instead use the geometry of the surface to check for stereo-consistency. Because apparent ridges shift geometric position with viewpoint changes, their screen-space projections may violate epipolar assumptions. Additionally, screen-space methods run into problems with occlusion, where features are partially matched when only partially visible in one view.

While prior work has addressed stereo-consistent rendering of suggestive contours and occluding contours, other view-dependent lines such as apparent ridges have not been explicitly studied in the context of stereoscopic rendering. Apparent ridges present unique challenges: they are interior surface features that shift position and are more sensitive to viewpoint changes than suggestive contours, making both screen-space epipolar matching and contour-based world-space matching unsuitable.

Additionally, none of the existing methods extend beyond line matching to address broader perceptual challenges in stereo line drawing, such as using hatching or shading methods to enhance depth and shape perception. Our work addresses both gaps: we develop a world-space correspondence method specifically designed for view-dependent interior surface features like apparent ridges, and extend it with hatching to provide stronger stereo depth cues.

2.7. Virtual Reality

Virtual reality (VR) is a simulated experience that places users in immersive, computer-generated environments. By providing an interactive, three-dimensional space, VR enables users to engage with digital worlds as if they were physically present [4]. This technology combines hardware and software components to deliver realistic visual, auditory, and sometimes haptic feedback, making it suitable for entertainment, training, education, and scientific visualization. In this paper, we only consider fully immersive VR: Systems that rely on head-mounted displays (HMDs) and motion tracking to create a complete sense of immersion.

2.7.1. Real time rendering in VR

In the context of real-time rendering, Virtual Reality presents unique challenges. Stereo-coherency between views to avoid visual discomfort, maintaining performance with multiple render passes due to the two cameras, and ensuring proper visual cues for accurate depth estimation. One of the most critical metrics in VR is motion-to-photon (MTP) latency, the delay between a user's physical movement and the update to the display. Higher latencies can lead to disconnect between the visual system and the physical body, resulting in simulator sickness [27, 25]. For a comfortable experience, this latency should be kept under 20ms [27, 20]. Additionally, according to LaValle [27] VR applications should render at consistent frame rates exceeding 90 frames per second for an immersive experience. These requirement places an upper bound on the computational budget of any line-drawing algorithm for VR.

2.7.2. Stereo-coherency in VR

Achieving stereo-coherency within any stereo environments have proven to be challenge, but is even harder to achieve with the additional computational budget. Any discrepancies between the stereo views, such as naive approaches described by Kim et al. [23] with view-dependent lines appearing at different orientations, results in binocular rivalry. Furthermore, even view-independent lines can cause binocular rivalry due to different occlusions in stereo views [15]. Thus, any real-time line drawing approach for VR must simultaneously satisfy both constraints: ensuring stereo-coherency to prevent binocular rivalry, while remaining with the tight computational budget imposed by VR latency requirements.

3

Methodology

In this chapter we describe the techniques and methodology of creating stereo-consistent apparent ridge hatching lines, as well as the motivation of the decisions made during the design. Results on how well this new method, stereo-consistent apparent hatching, performs will be described in chapter 4. The main contributions described are the following:

- A novel gradient-based search direction that, for any surface point, indicates the direction of steepest increase toward an apparent ridge.
- An iterative path-finding algorithm that traces and marks surfaces geodesically using the proposed derivative.
- A three view correspondence method in world-space for matching view-dependent lines across multiple viewpoints, enabling stereo-consistent line placement and increased depth perception.
- A hatching-based shading technique that generates stereo-consistent hatching lines from the path of marked surfaces.
- A controlled VR perceptual experiment evaluating the effectiveness of the proposed stereo-consistent hatching technique compared to standard AR renderings.

3.1. Stereo-consistent correspondence via gradient-based geodesic search

Matching view-dependent features across stereo viewpoints presents a major challenge: features that appear at different geometric locations depending on viewing angle can not be matched reliably using traditional epipolar geometry. We propose a gradient-based geodesic search algorithm that establishes correspondences by walking along the surface gradient field of any view-dependent scalar function. While the method can be applied to other feature lines, in this thesis we apply it specifically to apparent ridges, whose view-dependent curvature scalars are used to construct the gradient field.

3.1.1. Algorithm overview

Our algorithm establishes stereo correspondences for view-dependent features using a three-view sweeping approach. The core idea is to identify all faces on the mesh that contain a feature as the viewpoint transitions from center to stereo views. Rather than sampling intermediate viewpoints (which is computationally prohibitive), we efficiently approximate this sweep by performing gradient-guided geodesic walks on the mesh surface. Given a center viewpoint and left/right stereo views, we:

1. **Compute feature locations** for all three views by evaluating a view-dependent scalar field $f_v(p)$ at each face on the mesh, where v denotes the viewing position of each view.
2. **Precompute gradient fields** $\nabla f_v(p)$ for the left and right views across all mesh faces. These gradients point toward the nearest feature location for each respective view and guide the subsequent geodesic walks.

3. **Mark feature faces** where the scalar field satisfies a feature criterion (e.g., $f_v(p) = 0$ for apparent ridges).
4. **Sweep from center to stereo views**: For each feature face in the center view, we perform simultaneous geodesic walks toward the left and right views by following the precomputed gradient field $\nabla f_v(p)$ of left and right view on the mesh surface. This walk traces the path along which the feature moves across the surface as the viewpoint transitions from center to each stereo view.
5. **Mark traversed faces**: All faces visited during these geodesic walks are marked as part of the feature correspondence region. This set of faces represents an approximation of all locations where the feature would appear if we had sampled viewpoints between center and stereo views.
6. **Establish correspondence**: When a geodesic walk reaches a marked feature face in both the left and right views, we record a valid stereo correspondence linking the center feature to its left and right counterparts, otherwise the marked faces of this geodesic walk are discarded.

By anchoring searches from the center view and sweeping toward stereo views, we eliminate view bias and increase the likelihood that matched features belong to the same geometric structure. Figure 3.1 illustrates this sweeping concept: the coloured vectors on the circle shows how apparent ridge positions transition across the surface from the center view (AR_C) to the left view (AR_L, red arrow) and right view (AR_R, green arrow), while the vectors on the bottom between C and L and R denote the corresponding viewing positions.

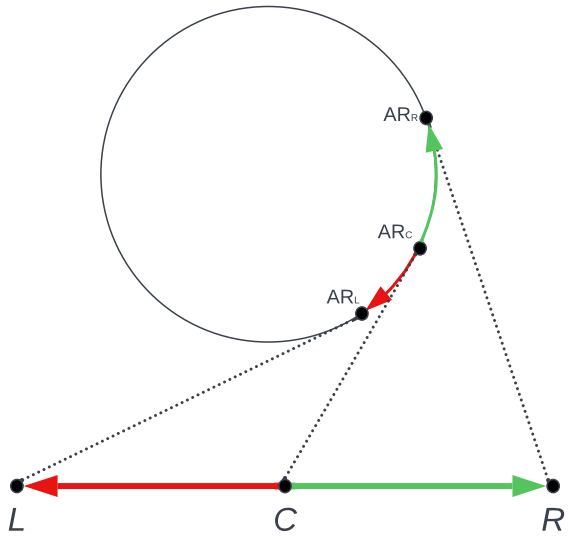


Figure 3.1: Visualization of the geodesic sweep from center to stereo views. The coloured curves across the sphere show how apparent ridge locations shift from the center view to the left view (red) and the center view to the right view (green).

3.1.2. Mathematical formulation

To perform geodesic walks on the mesh surface, we require a method to project any point towards the nearest feature location. We define a view-dependent scalar field $f(p)$ on the mesh, where the viewing position v is fixed for a given walk. For apparent ridges specifically, we use $f(p) = D_{t_1} q_1(p)$ Equation 2.3, where D_{t_1} is the directional derivative along the principal curvature direction $t_1(p)$ and $q_1(p)$ is the maximum principal curvature at point p [21]. Apparent ridges exist where $f(p) = 0$. More generally, this formulation applies to any view-dependent scalar field where features are characterized by zero-crossings or threshold criteria.

For an arbitrary point p where $f(p) \neq 0$, we seek a displacement vector Δp that points toward the nearest location satisfying the feature condition:

$$f(\mathbf{p} + \Delta\mathbf{p}) = 0 \tag{3.1}$$

Solving equation 3.1 precisely over the manifold is computationally expensive. Instead, we use a first-order Taylor expansion:

$$f(\mathbf{p}) + \nabla f(\mathbf{p}) \cdot \Delta\mathbf{p} \approx 0 \tag{3.2}$$

This simplifies to the linear constraint:

$$\nabla f(\mathbf{p}) \cdot \Delta\mathbf{p} = -f(\mathbf{p}) \tag{3.3}$$

Infinitely many vectors Δp satisfy equation 3.3. To find the nearest feature, we find the minimum-norm solution, which occurs when Δp is parallel to $\nabla f(\mathbf{p})$:

$$\Delta\mathbf{p} = k\nabla f(\mathbf{p}) \tag{3.4}$$

Substituting equation 3.4 into equation 3.3 and solving for k yields:

$$k = \frac{-f(\mathbf{p})}{||\nabla f(\mathbf{p})||^2} \tag{3.5}$$

This step size k determines how far to move along the gradient toward the feature, but also whether we should move in the positive or negative direction of the gradient. However, since the Taylor expansion is a first-order approximation, a single step rarely lands exactly on the feature. We therefore apply this iteratively: starting from an initial face, we check the pre-computed $\Delta\mathbf{p}$, move to the adjacent face along this direction, and repeat until reaching a marked feature face or exceeding a maximum step count.

While the scalar field $f(p)$ itself is view-dependent, the gradient $\nabla f(\mathbf{p})$ provides a stable direction field on the mesh for a fixed viewing position. This allows our algorithm to perform geodesic walks that converge to the nearest feature location for each specific view, ensuring geometric consistency across stereo pairs.

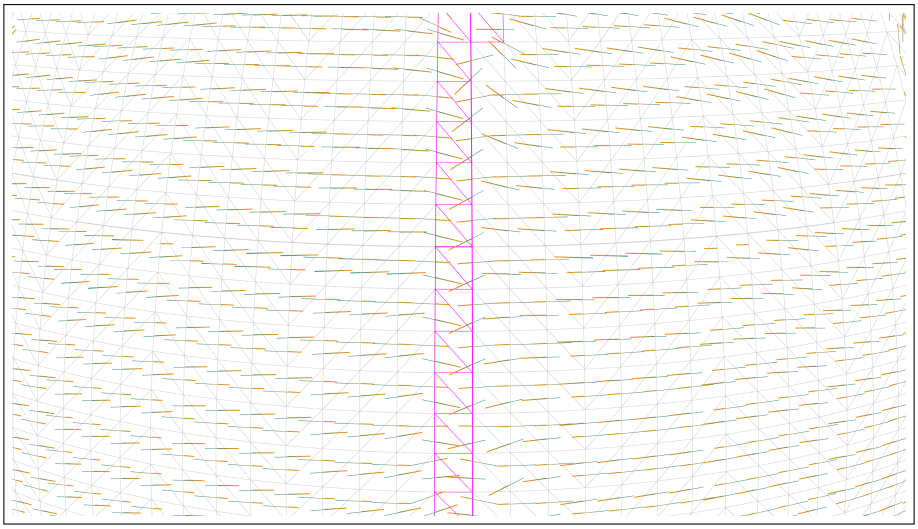


Figure 3.2: Visualization of the gradient field $\nabla f(\mathbf{p})$ pointing toward apparent ridge features (shown in magenta). Each face's gradient direction guides the geodesic walk toward the nearest ridge.

3.1.3. Three view correspondence algorithm

Using the gradient-based projection derived in Section 3.1.2, we implement the full correspondence algorithm detailed in Algorithm 1. The algorithm begins by precomputing the gradient field $\nabla f_v(p)$ for each face on the mesh for both the left and right stereo views. This precomputation happens once

per frame and provides a direction field that guides geodesic walks toward the nearest feature in each view.

For each feature face marked in the center view, we perform simultaneous geodesic walks toward stereo views. For each feature, we initialize two starting positions p_L and p_R on the starting face. At each iteration, if the current face is not marked as a feature face in its respective view, we use the precomputed gradient of that view and step to an adjacent face along the direction of the displacement vector from Equation 3.4. These walks continue until both reach marked feature faces in their view, at which we record valid stereo correspondence, or reach a step limit. All faces traversed for valid stereo correspondences are marked as our correspondence region, approximating the feature as the viewpoint moves from center to stereo views.

Algorithm 1 Three view Correspondence Algorithm.

Require: CenterAR (Faces identified as AR in center view)
Ensure: MatchedPairs (List of valid L-R ridge correspondences)

```
MatchedPairs ← ∅
for each face ∈ CenterAR do
    p_L ← face, p_R ← face
    foundL ← false, foundR ← false
    for step = 1 to MAX_STEPS do
                                                        ▷ Left view
        if not foundL then
            if p_L ∈ LeftARFaces then
                foundL ← true
            else
                p_L ← StepAlongGradient(p_L, ∇f_L)
            end if
        end if
                                                        ▷ Right view
        if not foundR then
            if p_R ∈ RightARFaces then
                foundR ← true
            else
                p_R ← StepAlongGradient(p_R, ∇f_R)
            end if
        end if
        if foundL and foundR then
            break
        end if
    end for
                                                        ▷ Stereo match found
    if foundL and foundR then
        MatchedPairs.add(p_L, p_R)
    end if
end for
return MatchedPairs
```

3.1.4. Application to apparent ridges

We apply our gradient-based correspondence algorithm to apparent ridges using the scalar field $f(p) = D_{t_1} q_1(p)$, where D_{t_1} is the directional derivative along the principal curvature direction and $q_1(p)$ is the maximum principal curvature. We mark faces containing apparent ridges by evaluating $f(p) = 0$ across the mesh for each view: center, left, and right.

Applying Algorithm 1 to these marked apparent ridge faces establishes stereo correspondences. However, several correspondence cases can arise:

- The matched faces correspond to the same geometric ridge structure.

- Multiple center ridges converge to the same left or right ridge.
- Matched ridges have different lengths or prominence across views.
- No match exists because a ridge appears in one view but not another.
- No match exists because a ridge appears in neither stereo view.
- Matched ridges are occluded in one or both views.

Our algorithm handles most cases naturally: geodesic walks follow the surface geometry and typically converge to corresponding ridge structures. The maximum step limit prevents infinite searches when features are missing in one view. However, cases involving different ridge lengths, occlusion, or partial visibility can result in visually inconsistent line drawings that trigger binocular rivalry. Additionally, because multiple center ridge segments can converge to the same stereo ridge, matched apparent ridges may appear segmented or discontinuous from stereo viewpoints. To address these perceptual challenges of rendering individual apparent ridge curves and enhance stereo coherence, we render the traversed surface regions using hatching lines, as described in Section 3.2. These hatching lines provide stereo-consistent visualization of surface curvature that apparent ridges cannot capture.

3.2. Hatching lines for stereo-consistent rendering

The gradient-based correspondence algorithm described in Section 3.1.1 marks all faces traversed during geodesic walks from center to stereo views, creating correspondence regions that capture where apparent ridges exist across the viewpoint transition. Even though our algorithm establishes correspondences, the difference in viewpoint positions between stereo views can still cause perceptual issues. Apparent ridges may be occluded in one eye but visible in the other, or may have different lengths across views. Additionally, since multiple center ridges can match to the same stereo ridge, the matched apparent ridges may appear segmented in stereo views. In extreme cases, these differences and segmented lines trigger binocular rivalry. The most straightforward approach would be to render all marked faces as solid regions. This would naturally be stereo-consistent as both views have the same regions drawn and have no problems with segmented lines. However, this produces large, unwanted blobs, particularly near contours as shown in Figure 3.3. These blobs fail to provide meaningful shape and depth cues because they lack structure and orientation. Instead of enhancing depth perception, these solid regions create noise that hinders the perception of underlying surface geometry.

Figure 3.3: An extreme example of a blob forming along the contour of a sphere for the left viewpoint when drawing the whole path naively.

3.2.1. Hatching as a solution

To address these limitations, we render the marked face regions using hatching lines rather than solid regions. Hatching has proven effective at conveying surface shape and creating an illusion of volume, especially for complex convex shapes and smooth objects [38]. Rather than rendering each marked face as a solid region, we render oriented line patterns that follow the surface geometry.

For our method, we adopt a hatching approach using stripe patterns. Stripes are curves that follow a specific vector field and in our case, the principal curvature directions of the mesh [24]. Figure 3.4 shows the Stanford Bunny rendered using the stripe generation technique by Knöppel et al. [24]. These stripes, used for hatch shading, convey curvature gradients, surface orientation, and local surface variation that are not captured by a single apparent ridge curve or solid region. By rendering hatching lines across the marked face regions established by our geodesic walks, we create a naturally stereo-consistent representation of shape that improves stereo depth perception while maintaining coherency across both eyes.

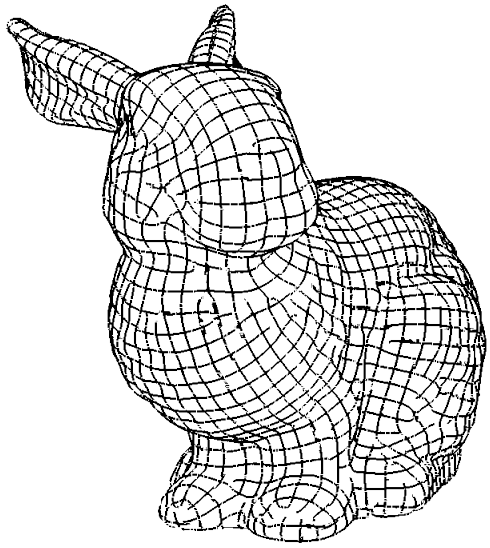


Figure 3.4: The stanford bunny, rendered with generated stripes from geometry central used for hatch shading.

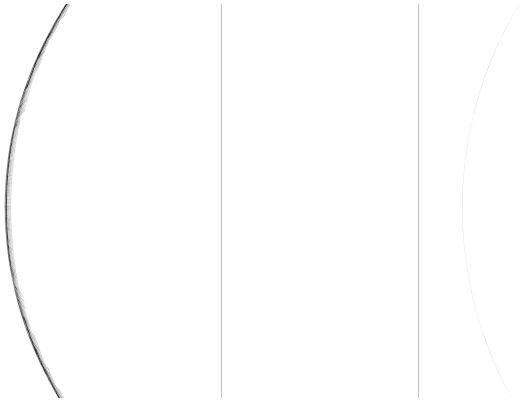
While the geodesic walks from center to stereo views establish correspondence regions, these regions are limited to the paths between explicitly matched apparent ridges. To further enhance depth perception and provide broader shape cues, we extend these paths beyond the matched ridge locations.

Once a geodesic walk reaches a matched apparent ridge face in both stereo views, we continue walking along the surface in the opposite direction of the original gradient field $-\nabla f(\mathbf{p})$. By moving opposite to the gradient direction, we grow the correspondence regions outward from the apparent ridges into surrounding areas of the surface. This can be seen as sweeping to viewpoints beyond our stereo viewpoints. Since the gradient magnitude vanishes at the apparent ridge locations (where $f(\mathbf{p}) = 0$), we use the gradient direction from the preceding walk steps to continue the path across and beyond the ridge.

This path extension serves two purposes. First, it enlarges the correspondence regions, dampening the impact of small view-dependent variations and providing more stable binocular depth cues. Second, it enlarges the region of stereo-consistent geometry surrounding each ridge that better convey surface structure around apparent ridges. Since these paths are extended from apparent ridges in world space, they are stable between left and right views, thereby reducing rivalry and improving overall depth and shape perception.

Figure 3.5 demonstrates the effect of path extension on a sphere. With extension (Figure 3.5a), the hatching bands are wider and provide clearer depth cues. Without extension (Figure 3.5b), the hatching is confined to narrower regions, reducing the effectiveness of the depth cues.

(a) Hatching lines on the sphere with extended paths for left and right viewpoints.



(b) Hatching lines on the sphere without extended paths for left and right viewpoints.

Figure 3.5: Comparison of extending and not extending paths beyond correspondence regions. Extended paths create broader hatching bands that provide stronger depth cues.

3.3. Implementation details

To evaluate the techniques described above, we developed an application that also served as the basis for the experiment discussed later in this thesis. The implementation is written in C++ and OpenGL. Two viewpoints are rendered using the off-axis projection method, as output for a VR headset and are displayed side-by-side on the computer monitor. Because all computations are performed in world space, producing the correct view for each eye only requires using the eye positions and applying the appropriate model–view–projection matrix.

The main steps of the implementation are:

- Loading a Wavefront .obj model.
- Computing per-vertex and per-face data for the mesh (curvature, normals, neighbourhoods).
- Generating or loading pre-computed stripes on the mesh with the stripe generation technique.
- Determining apparent ridge locations for the left, center, and right viewpoints.
- Calculating the gradient search direction per view for each vertex / face.
- Matching and extending apparent ridges from the center viewpoint to the left and right viewpoints.
- Rendering hatching lines along the matched face paths, or the apparent ridges on buffer textures.
- Rendering the buffer textures onto the VR headset and / or computer screen.

Mesh

Wavefront .obj files are imported using version 2.16 of Trimesh2, which expects the input to be a manifold triangle mesh, with per-vertex position data [33]. Trimesh2 handles the computations of normals,

neighbours, and principal curvatures, necessary for the computation of apparent ridges on the mesh. For each vertex, principal curvatures are calculated discretely, along with the principal curvature directions.

To ensure a direct mapping between the mesh manifold and the final rendering, we represent hatching lines as face-based polylines in world space. In this discretization, each mesh face contains a single linear segment that follows the local curvature field. These stripes are precomputed using the generation framework from geometry-central [37]. By having sets of stripes per face, the rendering process becomes a simpler lookup: once a face is marked, their associated stripes should be added to the draw buffer and be drawn. Multiple sets of stripes are generated and rotated to support cross-hatching. These sets are generated using varying frequencies to simulate different levels of detail, and approach inspired by the tonal art maps of Praun et al. [31]. To optimize performance and avoid repeated computation, each stripe set is exported to a separate file and re-used when needed. This ensures that users can switch between meshes quickly while avoiding the re-computation of stripe sets.

Shaders

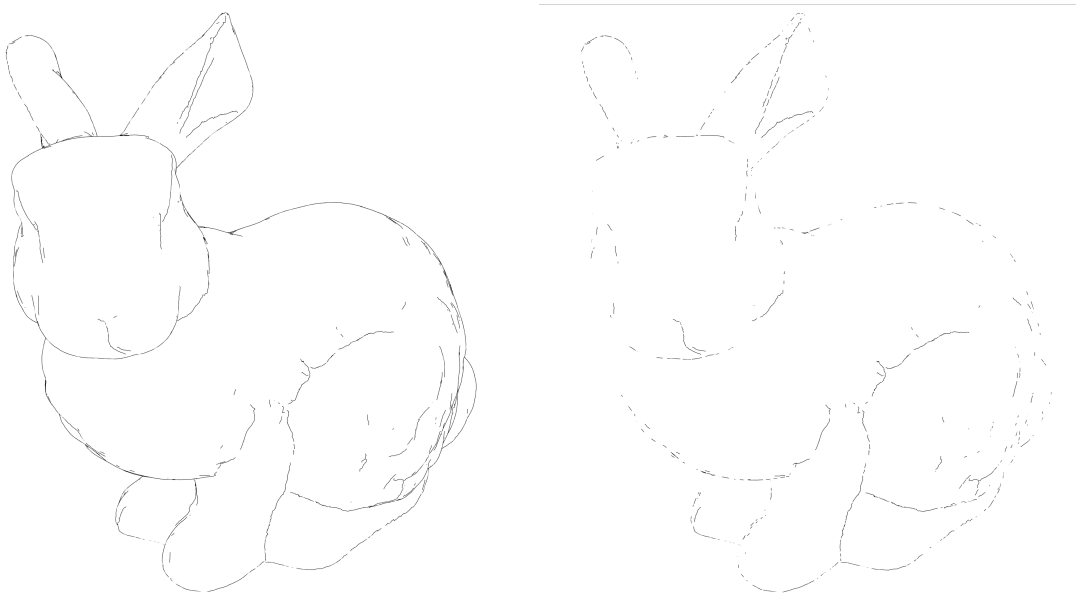
Compute shaders are used to compute basic view-dependent curvature information per-vertex and consequently used to calculate the apparent ridge positions and the gradient direction for each face for three viewpoints, left, center, and right. The computational code for apparent ridges is a modified GPU version of Judd, Durand, and Adelson [21], found in the RTSC viewer from Princeton. The main outputs we use are the indices of faces with an apparent ridge, the gradient direction per face, and the world-position of the apparent ridge line, used for the matching algorithm. Additionally, contour edges are calculated for each viewpoint in a separate compute shader, which use will be explained for the matching algorithm.

The main rendering shaders make use of the information of the compute shaders and matching algorithm to draw apparent ridge lines and hatching lines respectively. Apparent ridges are drawn using the apparent ridge positions calculated from the compute shaders. The hatching lines make use of the generated stripes to draw the correct stripes for each face marked by the matching algorithm. Since the position of the stripes are already computed in world-space, both eyes would see identical 3D lines, ensuring stereo-consistency. To prevent visual clutter, all lines can be rendered separately using a simple GUI.

Three view correspondence algorithm

The matching algorithm is executed on the CPU, operating on outputs of the compute shaders, specifically the apparent ridge face indices, the apparent ridge positions, and gradient directions of the left and right viewpoints for each vertex (∇f_L and ∇f_R). For each face containing an apparent ridge in the center view, the algorithm attempts to follow the gradient direction in both views from the middle position of the apparent ridge, performing a geodesic walk. The walk proceeds iteratively across the mesh. As the path intersects an edge, the algorithm samples the gradient direction by linearly interpolating the values stored at the edge's vertices. This ensures the accuracy of the search direction and that it remains continuous. This iterative walk proceeds until a corresponding apparent ridge is matched and extended beyond using the same amount of steps it took to reach the apparent ridge. This produces a set of paths, where paths originating from the same center apparent ridge that reached a match are considered valid.

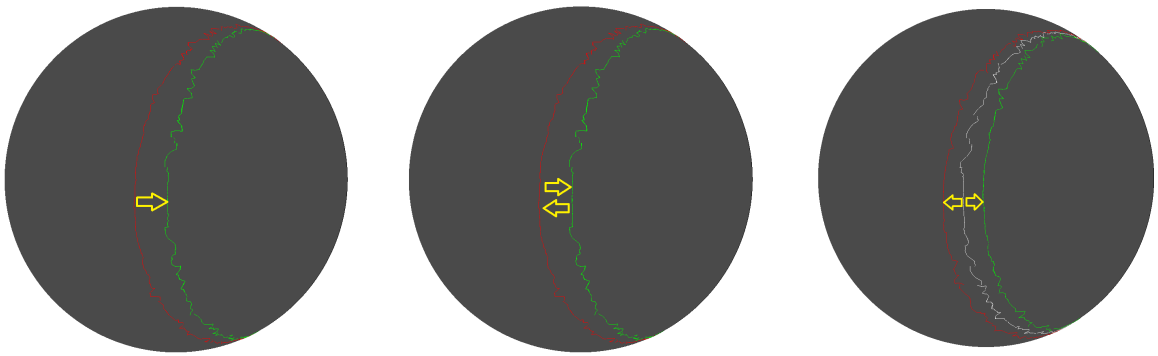
The reasoning behind the method we chose, is due to naive methods suffering from bias, inconsistencies, and less geometric accuracy. A simple algorithm for matching apparent ridge pairs would be to start from apparent ridges in either left or right view and find a match in the other view. While this method is capable of finding and matching apparent ridge pairs, it introduces directional bias towards one of the views, since of the viewpoints is treated as a starting point and thus 'ground truth'. The results might differ when changing which viewpoint to start with, as the gradient field is different for both viewpoints. Additionally, the matched apparent ridges in the other view appear fragmented due to the displacement vector pointing towards the closest apparent ridge as mentioned in Section 3.2. An example of this occurring can be found in Figure 3.6.



(a) Left-to-right matching on the Stanford Bunny for the left viewpoint.

(b) Right-to-left matching on the Stanford Bunny for the left viewpoint, apparent ridges appearing fragmented.

Figure 3.6: Illustration of the differences between apparent ridge lines for the naive methods left-to-right and right-to-left matching.



(a) Left-to-right matching.

(b) Left-to-right and right-to-left matching.

(c) Center-to-left and center-to-right matching.

Figure 3.7: Illustration of the three matching algorithms. Red lines correspond to the left viewpoint, white lines to the center viewpoint, and green lines to the right viewpoint.

To combat this, having both views as a starting point can resolve bias by only accepting pairs that exist in both. Despite there being no bias towards one view unlike the other naive method, this method still runs into problems such as false matches and finding the correct apparent ridge in the other view. Furthermore, due to the fragmented ridges that exist in either view, even less information would be shown and both viewpoints would appear fragmented. Finally, as this method requires us to perform the path search twice (L->R and R->L) this approach incurs additional overhead, undesired for real-time rendering. For these reasons, the method we adopt tries to avoid the additional overhead of searching twice, while maintaining no bias towards one of the views and avoiding the fragmented apparent ridges. For a better visual understanding of all methods, simple illustrations can be found in Figure 3.7

To ensure robustness against discretization artifacts and geometric edge cases, we implemented several safeguards and constraints within the matching framework. First, a search constraint is added in the form of a maximum step limit to mitigate computational overhead and prevent divergent searches. This prevents the algorithm from entering infinite loops in regions where the curvature field is poorly

defined, or in cases where a corresponding apparent ridge does not exist. Second we address the specific challenges posed by mesh discretization at contours. Theoretically, view-dependent curvature approaches infinity at these boundaries. On a discrete triangle mesh, this often results in fragmented ridge segments that are invisible when viewed in a standard render, but could break the continuity of a world-space geodesic walk. These fragmentations can cause our geodesic walk to 'miss' the apparent ridge resulting in a false negative. Furthermore, the gradient field near these boundaries are unstable as the view-dependent curvature nears infinity. We resolve these issues through two mechanisms: a pre-computation pass ensuring that the gradient direction is correct at these areas and the inclusion of contour edges. The pre-computation pass in a compute shader is performed by directing all the gradient directions of faces neighbouring an apparent ridge face towards the apparent ridge. In case of multiple apparent ridge face neighbours, the apparent ridge closest to the epipolar line is selected to preserve stereo-consistency. Contour edges are added to the correspondence algorithm, to prevent the false negatives of fragmented apparent ridges. Thus, faces containing a contour are also added and checked during the algorithm for correspondence: in case the geodesic walk hits a contour edge, it is considered valid as well.

To verify the physical plausibility of established correspondences, we apply a geometric distance threshold. By comparing the geodesic distances from the center apparent ridge to the left and right matches, we can identify false positives. If the disparity between these distances are significant, the match is discarded and not considered valid, as it likely represents a distant, unrelated feature found only due to the high step limit.

Finally, we handle degenerate cases where the center, left, and right ridge positions coincide on the same face. In such cases, the gradient direction is negligible, providing no correct directional information for the path extension as described in Section 3.2. If left unaddressed, these ridges would once again appear as thin lines providing minimal perceptual benefit. Thus, in these instances we symmetrically extend the hatching lines in directions perpendicular to the ridge tangent. This ensures that that even these ridges are transformed into stereo-consistent hatching surfaces that improve depth and shape perception. The length of this extension is a configurable parameter, allowing the amount of extended faces to be tuned per mesh.

Virtual Reality integration and implementation

The stereoscopic rendering and connecting the application to the HMD is done using Meta Horizon Link for connecting the HMD to the computer. Connecting the application to the headset was done using the OpenVR API, along with SteamVR as the runtime environment. These allow our application to access several components of the headset, such as eye position and projection data necessary for our three view correspondence algorithm.

The integration involves several technical aspects, most notably, the application retrieves the current HMD pose using the `WaitGetPoses` method for each frame. This method allows us to get the correct world position of each eye, as well as their respective model view projection matrices for rendering necessary for our three view correspondence algorithm. The center view anchor used in our three view correspondence algorithm is thus interpolated from these stereo world positions. Apart from this, we utilize additional Framebuffer Objects to capture left and right passes. Once the rendering of the hatching lines is complete, these textures are submitted to the SteamVR compositor, which handles final correction for the specific HMD. Additionally, as our application is calling the `WaitGetPoses` method each frame, the application is effectively synchronized to the specified refresh rate of the HMD. This causes our application to be synced with the refresh rate of the HMD, or halved in case the application is unable to reach the same FPS as the target refresh rate. In instances where the performance target is not met, the runtime halves the frame rate to maintain visual stability and initiates re-projection, a process creating synthetic intermediate frames by spatially shifting the previous frame for preserving the illusion of 90 FPS. Finally, as users usually have some head movement even when idle and the HMD is sensitive, the position of stereo viewpoints used for the calculation of apparent ridges and the gradient field shift often. This causes hatching lines to appear and disappear on surfaces even when the user attempts to sit completely still. To mitigate this effect, the stereo viewpoint positions are only updated if they cross a certain positional threshold, resulting in small movements still rendering the same hatching lines.

3.4. User study

To measure the efficacy of our gradient-based three view correspondence algorithm, we conduct a user study in VR. The goal of this study was to evaluate the perceptual efficacy of the proposed synthesized feature lines compared to traditional view-dependent apparent ridges. We specifically investigated how each method affects shape and depth perception, as well as visual comfort.

3.4.1. Hypothesis

The study was based on two primary hypotheses:

- **H1:** The synthesized feature lines provide more perceptual shape and depth cues than traditional apparent ridges.
- **H2:** Participants will perceive the synthesized lines as more stable, leading to increased viewing comfort with the proposed method.

3.4.2. Participants

A total of 12 participants were recruited for this study, approved by the HREC of Delft University of Technology with application ID 6207. All participants were screened for stereo blindness using a Randot Stereo Test and provided informed consent. Prior experience with VR was not necessary and thus not used as an exclusion criterion.

3.4.3. Apparatus and stimuli

Experiments were conducted using a Meta Quest 3 running at a refresh rate of 90 Hz. The VR headset was connected with a USB-C cable to a computer with a **AMD Ryzen 7 7800X3D**, and an **Nvidia GeForce RTX 4070 Ti Super**.

Stimuli

The experiment used 11 3D models with varying geometric properties and complexity:

Mesh	Source
Bunny	Stanford University Computer Graphics Laboratory
Dragon	Stanford University Computer Graphics Laboratory
Hat	Google LLC (2020), CC BY 4.0
Homer	Taken from https://github.com/alecjacobson/common-3d-test-models
Teapot	McGuire Archive, Martin Newell. Public Domain (CC0)
Yeah Right	MKeenan Crane, used under CC0 1.0
Shark	Google LLC (2020), Licensed under Creative Commons Attribution 4.0 International.
Tablecloth	Google LLC (2020), Licensed under Creative Commons Attribution 4.0 International.
Heptoroid	Carlo H. Séquin and Brent Collins, UC Berkeley.
Origins of the Pig	Keenan Crane, used under CC0 1.0
Blub	Keenan Crane, used under CC0 1.0

Table 3.1: Summary of 3D mesh stimuli used in the experiments.

3.4.4. Experiment procedure

The experiment followed a within-subjects design with three primary tasks. To eliminate order-bias and reduce the learning effect, rendering conditions and the order of the meshes are all randomized for each participant. For each task, Two-Alternative Forced Choice (2AFC) was used to evaluate the efficacy of our proposed method, the hatching lines, compared to traditional apparent ridges. An illustration of what the participant sees for Task 1 and 3 can be found in Figure 3.8 and Task 2 can be seen in Figure 3.9.

Task 1: Subjective shape and depth evaluation (H1)

Participants are asked to select which of the two renderings, apparent ridges or the synthesized hatching lines, they believe increases their shape and depth perception of 8 different meshes. Participants can switch between which set of lines they see. They are asked per mesh to select which rendering provided a clearer understanding of the shape and depth of the object.

Task 2: Relief matching (H1)
In task 2, participants are presented 2 distorted meshes with a randomized rendering, along with a third distorted reference mesh, corresponding to one of the two, rendered normally. These distortions are generated by displacing vertex positions along their surface normal using Perlin noise. To ensure a variety of stimuli, four different base meshes are each deformed four different times using anisotropic noise to create distinct geometric variations.

The reference mesh is rendered using a wood texture without any feature lines, while the two candidate meshes are rendered using either the proposed hatching method or traditional apparent ridges. In total, participants perform 8 trials, where they must select the 'correct' mesh that matches the reference mesh.

Task 3: Subjective viewing comfort evaluation (H2)
Finally, participants are asked which rendering they prefer for viewing comfort of the same 8 meshes as part 1. Participants can once again switch between which set of lines they can see, and are asked to select the rendering that is more comfortable. This comfortability should stem from less binocular rivalry and increased temporal coherence.

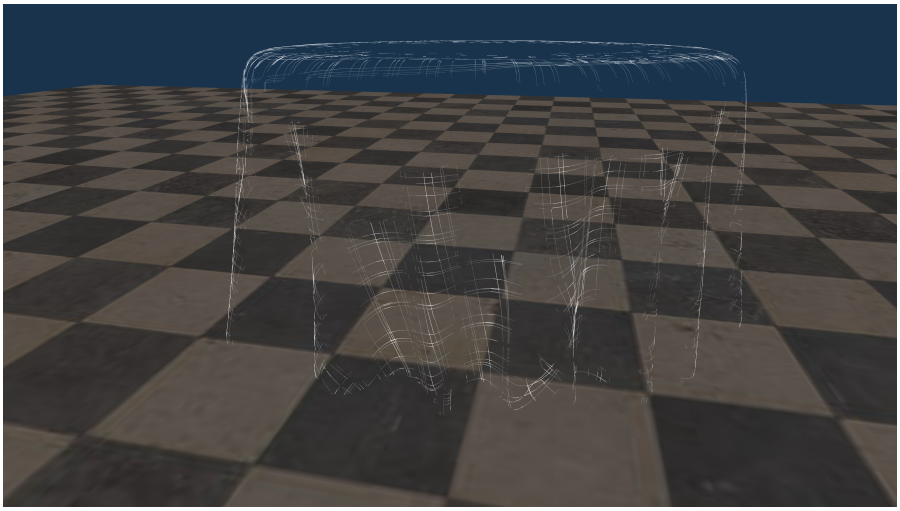


Figure 3.8: Example of tasks 1 and 3, hatching lines on the tablecloth mesh stimulus.

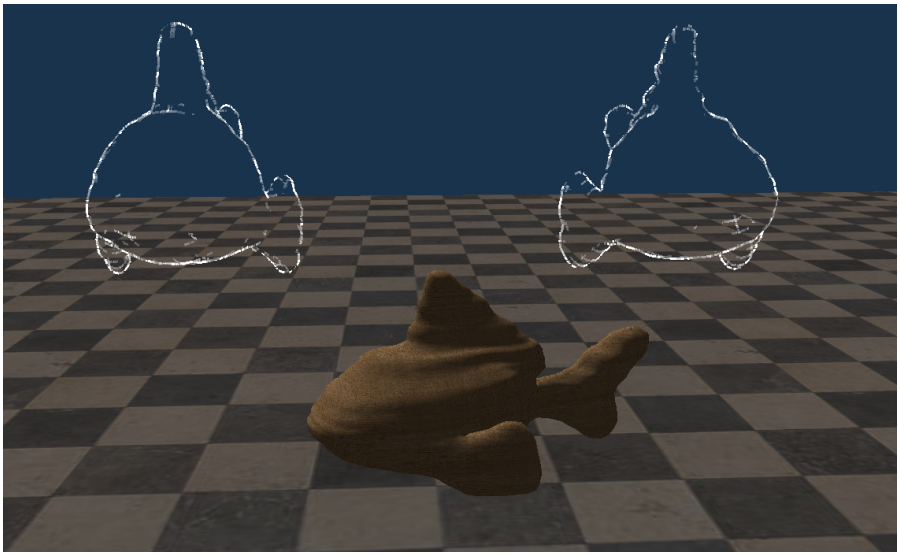


Figure 3.9: Example of task 2, distorted blub meshes with hatching lines and a reference wooden mesh under.

4

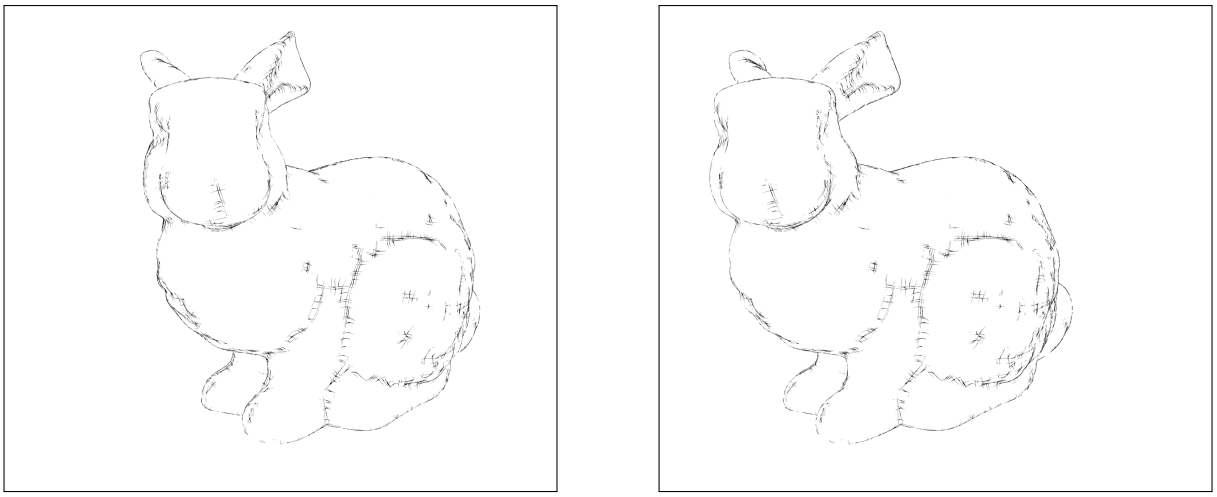
Results

This chapter presents the findings of this research in three parts. First, Section 4.1 presents the synthesized hatching lines, demonstrating how the three view correspondence algorithm shows various geometric features and viewpoints. Second, Section 4.2 provides a comprehensive gallery of the proposed method compared to the baseline applied to the full suite of meshes used during the experiment. Finally, Section 4.3 details the quantitative findings from the user study, evaluating the perceived efficacy of the hatching lines compared to traditional central apparent ridges regarding depth, shape perception, and user comfort within a VR environment. The primary goal was to determine whether these lines provide improved depth and shape cues compared to traditional center-viewpoint apparent ridges, in a VR environment.

4.1. Implementation results

The primary output of the proposed method is a set of view-dependent hatching lines. Figure 4.1 shows the lines on the Stanford Bunny.

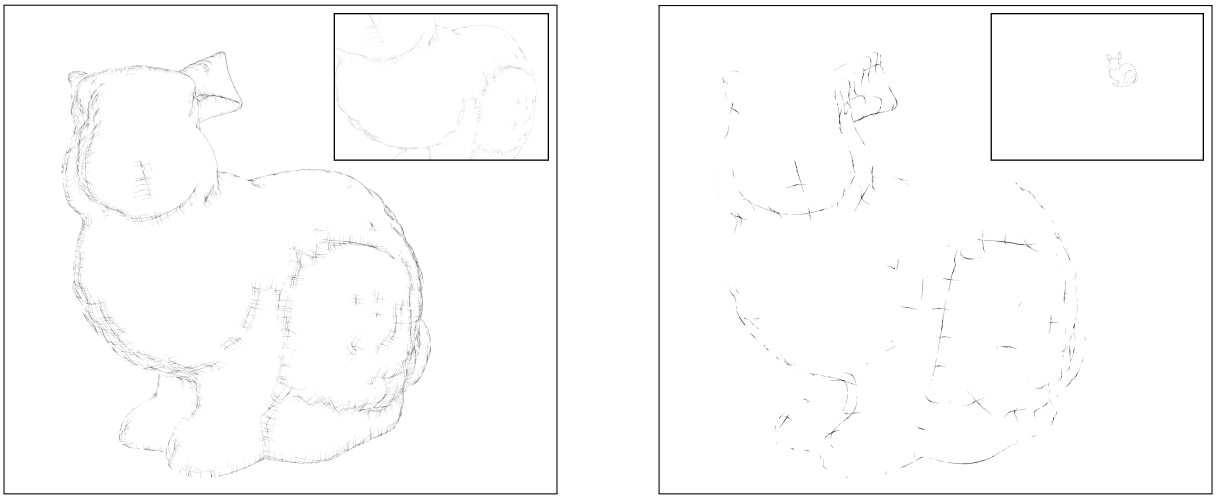
These hatching lines are based on the principal curvatures of the mesh, with varying line opacity. Line opacity changes the further away it is from the original apparent ridge (the center viewpoint). Additionally, the number of hatching lines and line thickness vary depending on the distance from the viewer to the mesh stimulus, based on the size of the mesh. Close distances would allow for thinner and more lines to be viewed when compared to far distances, as can be seen in Figure 4.2.



(a) Left viewpoint.

(b) Right viewpoint.

Figure 4.1: The final synthesized hatching lines of our method.

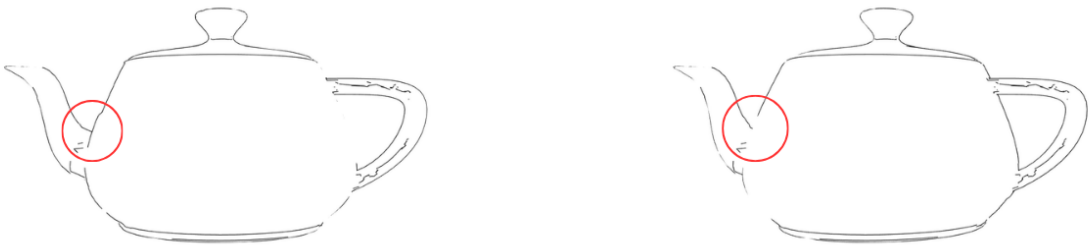


(a) Close distance.

(b) Far distance.

Figure 4.2: The number and thickness of hatching lines varying depending on viewing distance.

By using a fixed system of hatching lines, we try to increase temporal coherency and minimize binocular rivalry. Small movements would not cause the lines to change as much for the viewer, as the the lines already have a lower opacity on the end of the path. Additionally, the extension of the hatching lines beyond apparent ridges attempts to reduce binocular rivalry caused by occlusions. When compared to traditional apparent ridges rendered from a central viewpoint, differences can be seen when inconsistencies from occlusions are found. Figure 4.3 provides a direct comparison between traditional center-view apparent ridges (C-AR) and our proposed hatching method. In Figure 4.3a, the small differences in occlusion between viewpoints cause line segments to be visible in the left eye, but not the right. This is especially noticeable near the teapot's spout. In contrast, Figure 4.3b demonstrates how our method does not have this inconsistency from the same viewing position. Because the hatching lines are matched and extended, they provide additional lines that are visible in both eyes. Despite not all lines being visible in both eyes, the additional hatching lines provide enough content for the brain to maintain a stable 3D image.

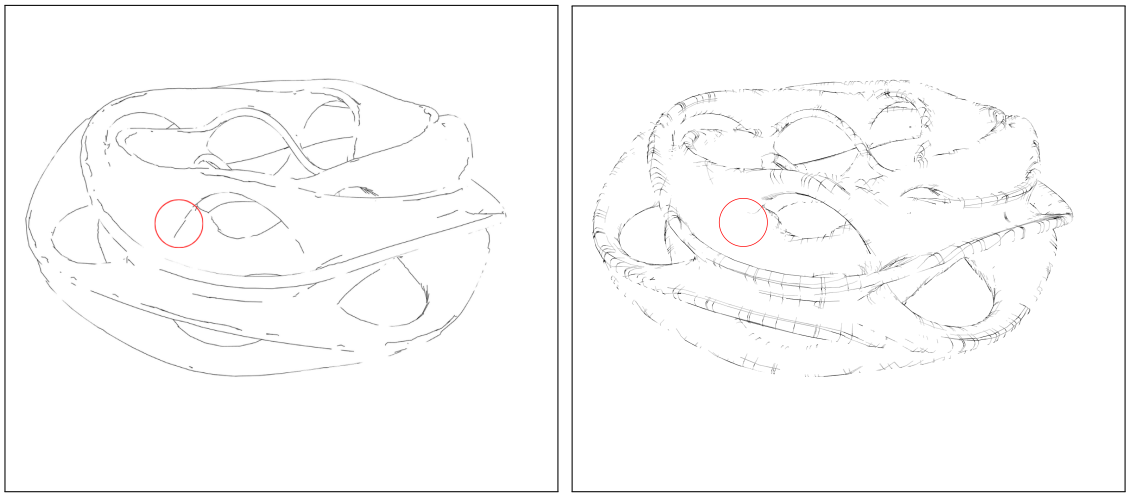


(a) Inconsistencies between viewpoints in the central apparent ridges on the teapot.

(b) No inconsistencies in our method, the hatching lines, on the teapot.

Figure 4.3: Comparison of viewpoint inconsistencies between central apparent ridges and our hatching method on the teapot.

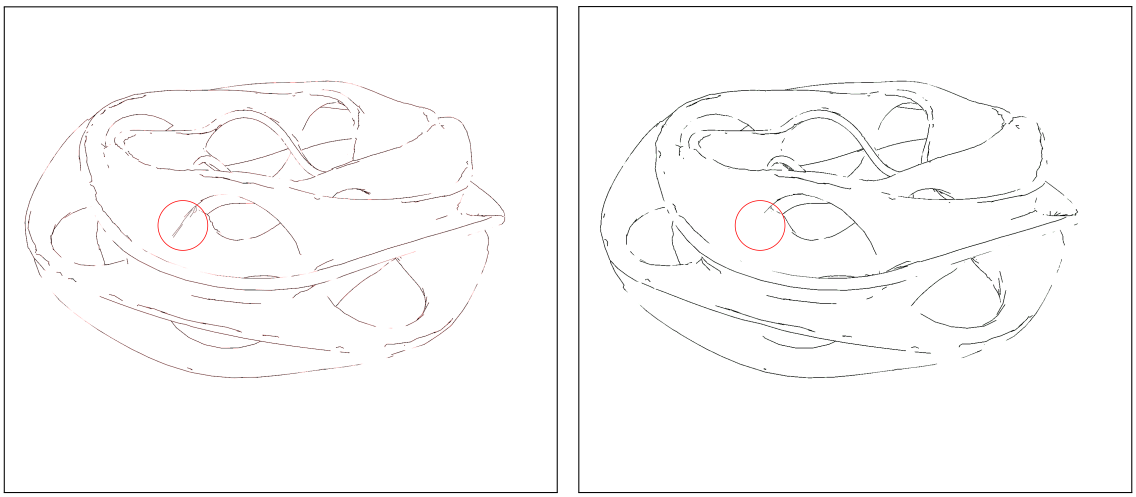
Since the matching algorithm attempts to create a match with apparent ridges from the center viewpoint to stereo viewpoints, some lines may be omitted when rendering the hatching lines compared to the central apparent ridges. This occurs when the pathfinding algorithm fails to find a valid apparent ridge in both views and has a similar effect as removing stereo-inconsistent lines in other stereo line drawing techniques. An example of this occurrence can be found in Figure 4.4, where lines are omitted in the hatching rendering. In this case the pathfinding algorithm fails to find a valid match in the right eye, as no apparent ridge exists, as seen in the right viewpoint of Figure 4.5. This behaviour is intentional as no apparent ridges exist in the corresponding viewpoint to form a valid stereo pair. This omission ensures that only features capable of being correctly stereo fused by the human visual system are displayed.



(a) Central apparent ridges viewed from the left viewpoint.

(b) Our method viewed from the left viewpoint.

Figure 4.4: The omittance of lines in our hatching method.



(a) Apparent ridges from the left viewpoint.

(b) Apparent ridges from the right viewpoint.

Figure 4.5: The apparent ridges of a heptoroid, rendered from stereo viewpoints.

While the iterative pathfinding effectively synthesized consistent lines for the majority of surface geometry, specific failure cases were observed, particularly in regions of high view-dependency such as contours. As shown in Figure 4.6, false negatives can occur where the algorithm fails to establish a match between the central reference ridge and their stereo-view counterparts, despite that they should be matched. These omissions are most prevalent near the silhouettes or contours of the model, such as the left side of the Homer mesh's head.

In these regions the baseline center view apparent ridges are clearly capable of highlighting the silhouettes of the mesh, yet are omitted in the final render of the hatching lines. This results in a 'fragmented'

appearance where a continuous line in the apparent ridge method is represented as a series of disconnected hatching segments, despite our efforts of mitigating these false negatives by adding contour edges and redirecting the gradient direction.

(a) Central apparent ridges from the left viewpoint. **(b)** Hatching lines from the left viewpoint.

Figure 4.6: Failure of finding a match near contours on the Homer mesh.

4.1.1. Performance

Performance of the pathfinding algorithm and rendering of the hatching lines is assessed using a laptop with a **12th Gen Intel(R) Core(TM) i7-12700H** and **NVIDIA GeForce RTX 3070 Ti Laptop GPU**. Performance was measured using across two modes: a standard PC configuration at a resolution of 1920 $\times$ 1080 when not connected to the VR headset, and a VR configuration using a Meta Quest 3, with a resolution of 3712$\times$1984 running at 90 Hz.

Table 4.1 presents the rendering performance of all meshes used in the experiment, for both VR and non VR. Frame rate data was captured using CapFrameX [36], where the average FPS is measured over 10 seconds when running the application.

VR performance

While the system was capable of maintaining high performance in the PC configuration, complex meshes with high amounts of faces failed to meet the VR performance requirement of 11.1 ms or 90 FPS within the VR configuration. Most notably, 3 meshes were incapable of reaching the performance metric needed of 90 FPS within the VR environment and were instead sustained at 45 FPS. This behaviour is caused by the frames failing to align with the refresh cycles of the display. As detailed in Section 3.3, the application's rendering loop is synchronized with the refresh cycles of the HMD. When the application fails to meet the target refresh rate, the runtime automatically halves the frame rate to maintain visual stability and prevent stuttering, despite the hardware being capable of higher speeds such as the Bunny reaching 85.5 FPS.

4.2. Visual gallery

To demonstrate the proposed method with the baseline apparent ridges, this section provides a visual overview of the results across all mesh stimuli used for this Thesis. Figure 4.7 shows the various mesh stimuli with our proposed method.

4.3. Experiment results

The primary objective of the user study was to empirically validate the efficacy of the hatching lines compared to traditional center-viewpoint apparent ridges. The evaluation focuses on three tasks, shape and depth perception, relief matching, and viewing comfort. An overview results can be found in Table 4.3 and Figure 4.8.

Figure 4.7: Visual gallery of all mesh stimuli using hatching lines (left) and central apparent ridges (right).

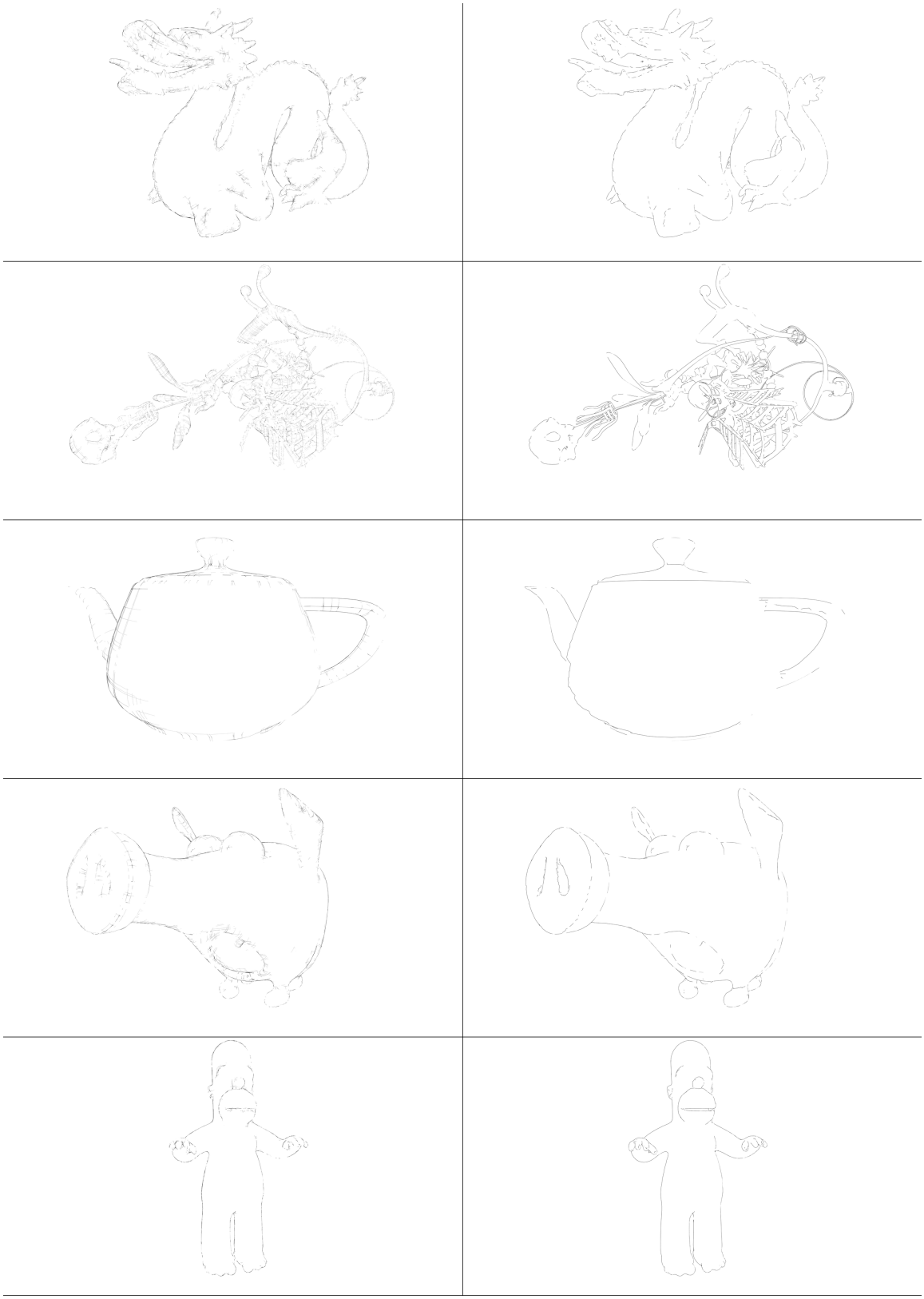


Figure 4.7: Visual gallery of all mesh stimuli using hatching lines (left) and central apparent ridges (right), (continued).

Mesh	Vertices	Faces	PC FPS	VR FPS
Bunny	72,027	144,046	124.7	85.5
Dragon	43,457	87,130	142.2	101.6
Hat	49,656	99,324	114.1	80.3
Homer	30,608	61,212	253.2	180.6
Teapot	7,850	15,704	645.3	372.7
Yeah Right	94,076	188,638	61.5	46.0
Shark	17,738	35,472	325.7	217.0
Tablecloth	39,291	78,144	166.1	123.2
Heptoroid	17,878	35,840	305.3	195.1
Pig	27,904	55,712	246.6	153.7
Blub	7,106	14,208	695.5	428.6

Table 4.1: Performance metrics for 3D mesh stimuli. PC FPS measured using a single view and VR FPS measured without synchronization.

4.3.1. Task 1: Shape and depth perception

Participants were asked to choose which rendering method, central apparent ridges or hatching lines, provided more shape and depth cues for the participant in Task 1. This was measured as a binary question, using 2AFC. The quantitative results found in Figure 4.8 Task 1, indicate a statistically significant preference for the proposed hatching method. A two-tailed binomial test with Bonferroni correction was conducted to determine if users had a statistically significant preference for hatching lines over the central apparent ridges. Results indicated that for 12 participants and 8 meshes, for 71 out of the 96 meshes participants preferred hatching lines (73.96%). Results indicate statistical significance, with a Bonferroni-corrected p-value of $0.0000229 < 0.05$. This preference was especially noticeable for certain meshes, where users would prefer the hatching method much more for the heptoroid (12/12), tablecloth (12/12), and the shark (11/12) as seen in Table 4.2.

Mesh	Hatching Preference	p-value
Bunny	7/12	0.7744
Dragon	10/12	0.0386
Heptoroid	12/12	0.0005
Homer	5/12	0.7744
Shark	11/12	0.0063
Tablecloth	12/12	0.0005
Teapot	7/12	0.7744
YeahRight	7/12	0.7744

Table 4.2: Per-mesh preference analysis for Part 1: Shape and depth perception.

4.3.2. Task 2: Relief matching

During Task 2, participants were shown three meshes, two distorted meshes and a relief corresponding to one of the two distorted meshes, across 8 different relief matches: 4 for apparent ridges and 4 for our proposed hatching lines. Participants are then tasked to select which of the two distorted meshes correspond to the relief. This was measured as a binary question, using 2AFC, and evaluated using a Chi-Square test. Results for the matching task can be found in Figure 4.8 Task 2. While results show that both methods allowed participants to perform well above the 50% chance level, there was no statistically significant difference in accuracy between the two methods. Participants answered correctly for 36/48 for the apparent ridge method and 34/48 for the hatching lines. Results of the Chi-Square test showed ${}^{2}(1, N = 96) = 0.05$ and a p-value of $0.8183 > 0.05$, indicating no statistical significance.

4.3.3. Task 3: Viewing comfort

Viewing comfort was evaluated using the same method as Task 1. Participants were asked to choose which rendering method was more comfortable to view for 8 meshes in Task 3. This was measured

as a binary question, using 2AFC, and a two-tailed binomial test with Bonferroni correction. Results show no statistical preference for viewing comfort between the two methods as can be seen in Figure 4.8 Task 3. Out of the 96 answers, 52 preferred hatching lines (54.17%). Results indicate no statistical significance, with a Bonferroni-corrected p-value of $1.000 > 0.05$.

Task	Hatching	Central AR	Test statistic	p-value
1. Shape & Depth	74.96%	26.04%	Binomial	0.0000229*
2. Relief Matching	70.83% Correct	75% Correct	Chi-Square ($X^2 = 0.05$)	0.8183
3. Viewing Comfort	54.17%	45.83%	Binomial	1.000

Table 4.3: Experiment results across tasks. *indicates statistical significance (p < 0.05).

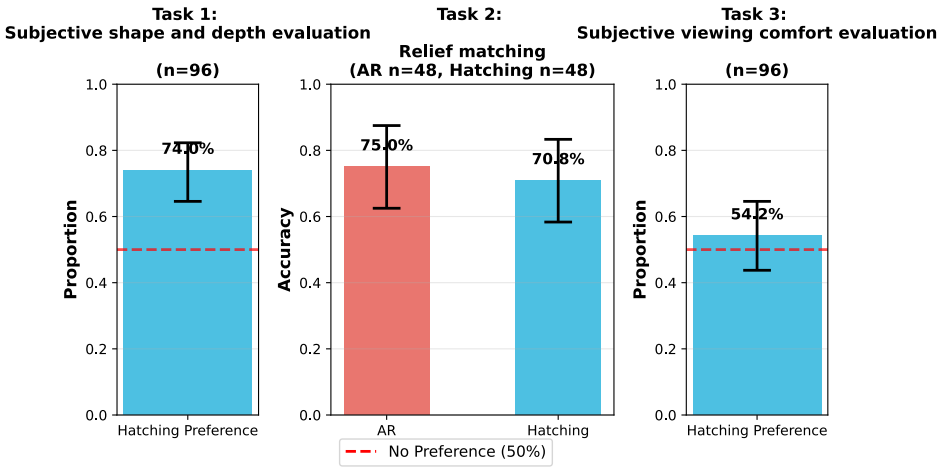


Figure 4.8: Results of the experiment with 95% Confidence Intervals.

5 Discussion

This chapter presents a discussion of the implementation and experimental results obtained in Chapter 4, as well as the limitations of our proposed method. Additionally, this chapter highlights several potential improvement directions for future work.

5.1. Discussion

5.1.1. Implementation

The implementation of the world-space three view correspondence algorithm successfully established correspondences between the reference central apparent ridges and stereo viewpoints. By tracing these paths directly on surface geometry, we moved beyond the image-space constraints of previous works [23, 18]. This ensured that the resulting hatching lines were naturally stereo-consistent, with improved depth and shape cues. However, the iterative nature of the pathfinding algorithm revealed specific sensitivities to the underlying mesh complexity and the gradient field.

The iterative pathfinding algorithm was unable to consistently maintain a valid path in regions of extreme view-dependency like contours. As noted in the Results 4.1, the Homer mesh contains 'fragmented' lines near contours. This failure is likely due to the sensitivity of the gradient field near contours. Theoretically, contours in apparent ridges are the loci of points where the view-dependent curvature goes to infinity. This is not possible to calculate on discrete meshes however, and results in unintended behaviour of the gradient field at or near contours. Near the contours, this gradient field can become noisy and unstable due to the mesh's triangulation. Despite our efforts to fix the gradient field near contours and include actual contour edges as a valid match, edge cases can occur where the iterative pathfinding algorithm is incapable of finding a valid mesh due to unintended gradient directions. In these cases, the path might repeat between two faces, reaching the maximum step limit set by the algorithm. This results in a false negative, where the ridges that are spatially close in world-space and theoretically should match are not successfully connected.

5.1.2. Performance and VR viability

The performance analysis found in Table 4.1 indicate both promises and challenges for our proposed method in VR environments. To accurately evaluate the performance of these meshes, FPS was measured without the `WaitGetPoses` call each frame, decoupling the rendering loop from the HMD's refresh rate to prevent the application from stalling and allow for more precise measurements. While the method successfully maintains the 90+ FPS requirement for most meshes (8 out of 11), three complex meshes: the Bunny, Hat, and the Yeah Right mesh fell short of this target, which would maintain 45 FPS throughout during actual usage. Consequently, this would mask true hardware capability, as evidenced by the Bunny mesh achieving a much higher performance of 85.5 FPS without synchronization.

The performance limitation stems primarily from the CPU-bound nature of the iterative pathfinding algorithm and the overhead associated with CPU-GPU data transfers and vice versa. Complex meshes with high face counts of approximately 100.000 faces or more, failed to hit the performance require-

ments, while simple meshes like the Teapot were consistently maintaining frame rates well above the threshold. This is due to the current implementation scaling with geometric complexity and number of faces. Higher number of faces would result in an increasing number of apparent ridge lines, as they are calculated per face, which results in more work for the CPU-bound pathfinding algorithm.

Despite not achieving 90 FPS across all test cases using the PC configuration, the method demonstrates practical viability for VR applications with moderately complex geometry. However, with a more powerful CPU such as the one used for the experimental study, 90 FPS was achieved throughout all meshes and tasks. For production deployment in other VR applications requiring high frame rates across arbitrary meshes, further optimisations would be necessary, to ensure that the performance requirements of VR are met consistently.

5.1.3. Experimental study

The user study results provided statistical evidence that the proposed method successfully addressed the main research question: *How can the synthesis of stereo-consistent lines inspired by line drawing techniques enhance the 3D depth and shape perception of users in Virtual Reality?*.

Task 1

The statistically significant preference for hatching lines in Task 1 (74.96%, p = 0.0000029) demonstrates that users perceive substantially improved depth and shape cues with the proposed method. The variation in preference across different meshes offer insights on when and why the proposed hatching lines excel. While most meshes have a more balanced preference, the heptoroid, tablecloth, and shark all have unanimous or near-unanimous preferences for the hatching lines (12/12, 12/12, and 11/12 respectively, from Table 4.2). This suggests that the method is particularly effective at conveying depth and shape cues for surfaces with complex varying curvature orientations, which traditional apparent ridges, being sparse and thin, struggle to convey. In contrast, the hatching lines provide information across the surface, making convex and concave regions more distinguishable.

Conversely, the balanced preference of other meshes might be harder to explain. Prior knowledge of familiar objects like the Teapot, Bunny, and Homer mesh might compensate for sparser line information. When users already possess strong mental imagery of the models shape, additional visual cues provide diminishing returns. Additionally, our proposed method is sometimes still incapable of capturing all hatching lines, especially near contours, causing users to experience missing or lacking lines with the hatching lines. This effect is further enhanced when the users already have a mental image of the shape in mind. Finally, the Yeah Right mesh, despite it's complexity, did not show any significant preference for either method. We believe this is due to the overly complex geometry, where the lighter opacity hatching lines introduce more visual clutter and make it harder to see interior silhouettes.

Task 2

The null result for Task 2's relief matching (p = 0.8183) does not necessarily indicate a failure of the hatching method, but rather highlights the challenge of designing matching tasks with appropriate difficulty levels. The task design struggled to find a balance between trivially easy matches, which renders the method irrelevant, and impossibly difficult, where even enhanced depth and shape cues cannot help. This suggests that the depth and shape benefits from Task 1 may lie in qualitative perception, rather than specific geometric discrimination tasks. Alternatively, these results may indicate that while our method does not offer a measurable advantage in geometric interpretation for these specific models, it similarly does not reduce performance compared to the baseline of central apparent ridges.

Task 3

The neutral result in Task 3 regarding viewing comfort is particularly significant, given the increased density of lines with the proposed method. This finding addresses a key concern whether the increased visual information for shape and depth cues provided by the hatching lines would decrease visual comfort. The fact that users did not find the denser hatching lines to be less visually comfortable, indicate that enhanced shape and depth cues may be more important as they do not hinder visual comfort.

On the other hand, despite our method demonstrating cases where occlusion-based inconsistencies are removed in Figure 4.3b, participants did not prefer the hatching method over traditional apparent

ridges. This could be due to the VR environment allowing users to rotate the mesh, where participants can miss these cases when constantly moving the mesh. Other reasons for why the proposed method is not preferred over traditional apparent ridges, could be attributed to stylization of the hatching lines and inconsistencies over the mesh. Users finding missing lines near contours, compared to the clear high opacity apparent ridge, can cause a preference for these apparent ridges when asked about visual comfort.

5.2. Limitations

While the proposed method demonstrate clear benefits for stereo-consistent line rendering in VR, several limitations warrant discussion.

Computational performance

The primary limitation is inability of the CPU-bound pathfinding algorithm to consistently reach the VR performance requirements for complex geometry on non high-end CPUs. The iterative nature of the matching algorithm, introduces significant computational overhead that scales poorly with mesh complexity. The CPU-GPU and GPU-CPU data transfers required further compounds this bottleneck. This limitation restricts the method to moderately complex scenes and prevents its use in applications requiring 90 FPS without a high-end CPU available. One solution would be to move the algorithm to the GPU, removing the overhead created from data transfers between CPU and GPU, removing the need for a high-end CPU.

Pathfinding logic

Another notable limitation of the current implementation is its susceptibility to failure in regions near the contours. The definition of contours in apparent ridges causes issues for the gradient field due to mesh triangulation resulting in false negatives. In such cases, the algorithm is incapable of finding a valid stereo match, causing a 'fragmented' look for lines that are supposed to look continuous.

While these 'fragmented' lines are stereo-consistent, it creates a lack of visual completeness. These false negatives are particularly jarring on smooth, well known meshes such as the Homer mesh, where users expect a continuous silhouette to define the boundary of the shape.

Geometric dependency

The effectiveness of this method varies considerably with surface geometry. While it has shown to excel for meshes with clear surface variations, extremely complex geometry may not benefit from the additional information. On the other hand, the hatching lines are drawn on the path of faces found by the iterative pathfinding algorithm, which are triangles, overly simple meshes may show extreme triangular patterns of the drawn hatching lines. Additionally, the current implementation is only capable of handling triangle meshes, limiting the number of applications for our method.

Limited experimental scope

The user study evaluated 12 participants across 8 meshes in a controlled environment. While this sample size was enough to detect statistically significant differences in Task 1, the generalizability of these findings to different mesh types or varied application tasks remain unclear. Furthermore, the failure to find significant results in Task 2 may reflect experimental design limitations, rather than true equivalence between the two methods.

Temporal coherence

While the implementation attempts to address temporal coherence through a fixed system of hatching lines with stylization, the experiment did not explicitly measure temporal consistency. The assumption that the method improves temporal coherency, while theoretically possible, lacks empirical validation from the user study.

5.3. Future work

Several promising directions could address the current limitations of the proposed method and extend the capabilities of stereo-consistent hatching lines in VR.

One of the most critical improvement would be reimplementing the iterative pathfinding algorithm entirely on the GPU using compute shaders and taking into account further edge cases. **GPU-based pathfinding** would leverage the parallelization capabilities that the GPU has, as well as eliminate data transfer overhead between the CPU and GPU. While GPU pathfinding is much more difficult than CPU pathfinding, having GPU pathfinding for the proposed method could potentially reduce computation time by orders of magnitude, enabling consistent 90 FPS performance even for complex geometry and across lower end systems. Furthermore, removing the false negative cases that the current implementation has, would improve visual clarity and possibly visual comfort. Creating hatching lines that are consistently continuous across the surface boundary would necessitate a more robust path-traversal logic. This way, the method could maintain stereo-consistent lines without sacrificing the visual integrity of silhouettes.

Other improvements can be found in **optimising temporal coherence**. While the current implementation does address temporal coherence, explicit tracking of apparent ridges or the rendered hatching lines across frames could further improve stability. Maintaining line correspondence across frames could reduce temporal flickering due to viewpoint changes.

Furthermore, the current method uses an **adaptive level of detail** for line density as inspired by Tonal Art Maps [38], by generating each level of detail individually. While this may work for certain meshes, it is much more preferred to generate one high density level of hatching lines and creating the lower levels by culling lines from the level above. This results in more evenly spread hatching lines across viewing distance and mesh complexity.

Finally, **extending the perceptual evaluation** needs to be considered. Future studies should include comparative evaluation against other stereo line drawing methods, possibly through continuous depth and shape estimation rather than binary preferences. As well as extending the study to other context such as medical visualization, CAD review, or educational applications, with better designed tasks capable of measuring objective depth and shape perception.

6

Conclusion

This thesis investigated how stereo-consistent hatching lines derived from apparent ridges can enhance depth and shape perception in Virtual Reality. By implementing a gradient-based geodesic search algorithm that operates in world-space, we addressed the binocular rivalry and visual instability associated with view-dependent line rendering in stereoscopic environments.

The findings of this research lead to three primary conclusions. First, dense hatching lines based on principal curvatures provide stronger depth and shape cues compared to sparse apparent ridge curves alone. This improvement is particularly evident for complex, unfamiliar geometries where individual apparent ridges cannot fully convey surface structure as seen from the experiment results. Second, anchoring correspondence searches at a center viewpoint and performing geodesic walks toward stereo views ensures geometric consistency between left and right eyes. Our user study confirmed that this approach maintains viewing comfort while providing enhanced visual information. Third, despite the perceptual benefits of the proposed method, the current CPU-bound implementation demonstrates the demanding performance requirements of real-time VR rendering.

In summary, this work demonstrates that hatching-based line rendering can serve as a functional tool for effectively communicating shape and depth in VR. The gradient-based correspondence method establishes a foundation for stereo-consistent rendering of view-dependent features in world-space. We hope this work encourages further research in real-time non-photorealistic rendering for Virtual Reality, particularly in developing methods that balance perceptual effectiveness with the strict performance constraints of immersive environments.

References

[1] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/crc Press, 2019.

[2] David Alais and Randolph Blake. *Binocular rivalry*. MIT press, 2005.

[3] Martin S Banks et al. “Stereoscopy and the human visual system”. In: *SMPTE motion imaging journal* 121.4 (2012), pp. 24–43.

[4] Woodrow Barfield et al. “Presence and performance within virtual environments”. In: *Virtual environments and advanced interface design* (1995), pp. 473–513.

[5] Boubakeur Boufama and Roger Mohr. “Epipole and fundamental matrix estimation using virtual parallax”. In: *Proceedings of IEEE International Conference on Computer Vision*. IEEE. 1995, pp. 1030–1036.

[6] Dennis R Bukenberger, Katharina Schwarz, and Hendrik PA Lensch. “Stereo-Consistent Contours in Object Space”. In: *Computer Graphics Forum*. Vol. 37. 1. Wiley Online Library. 2018, pp. 301–312.

[7] Adrien Chopin, Daphne Bavelier, and Dennis Michael Levi. “The prevalence and diagnosis of ‘stereoblindness’ in adults less than 60 years of age: a best evidence synthesis”. In: *Ophthalmic and Physiological Optics* 39.2 (2019), pp. 66–85.

[8] ANM Imroz Choudhury and Steven G Parker. “Ray tracing NPR-style feature lines”. In: *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*. 2009, pp. 5–14.

[9] Forrester Cole et al. “How well do line drawings depict shape?” In: *ACM SIGGRAPH 2009 papers*. 2009, pp. 1–9.

[10] Forrester Cole et al. “Where do people draw lines?” In: *ACM SIGGRAPH 2008 papers*. 2008, pp. 1–11.

[11] Doug DeCarlo, 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*. 2004, pp. 15–145.

[12] Doug DeCarlo and Szymon Rusinkiewicz. “Highlight lines for conveying shape”. In: *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*. 2007, pp. 63–70.

[13] Doug DeCarlo et al. “Suggestive contours for conveying shape”. In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, pp. 401–408.

[14] Olivier D Faugeras. “What can be seen in three dimensions with an uncalibrated stereo rig?” In: *European conference on computer vision*. Springer. 1992, pp. 563–578.

[15] Davi Geiger, Bruce Ladendorf, and Alan Yuille. “Occlusions and binocular stereo”. In: *International Journal of Computer Vision* 14.3 (1995), pp. 211–226.

[16] Robert M Haralick. “Ridges and valleys on digital images”. In: *Computer vision, graphics, and image processing* 22.1 (1983), pp. 28–38.

[17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] Dejing He, Rui Wang, and Hujun Bao. “Real-time rendering of stereo-consistent contours”. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 81–87.

[19] Aaron Hertzmann and Denis Zorin. “Illustrating smooth surfaces”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 517–526.

[20] Jason Jerald. *The VR book: Human-centered design for virtual reality*. Morgan & Claypool, 2015.

[21] Tilke Judd, Frédo Durand, and Edward Adelson. “Apparent ridges for line drawing”. In: *ACM transactions on graphics (TOG)* 26.3 (2007), 19–es.

[22] Bela Julesz. “Stereoscopic vision”. In: *Vision Research* 26.9 (1986), pp. 1601–1612.

[23] Yongjin Kim et al. “Stereoscopic 3D line drawing”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–13.

[24] Felix Knöppel et al. “Stripe patterns on surfaces”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–11.

[25] Eugenia M Kolasinski. “Simulator sickness in virtual environments”. In: (1995).

[26] Manuel Lang et al. “Practical temporal consistency for image-based graphics applications”. In: *ACM Transactions on Graphics (ToG)* 31.4 (2012), pp. 1–8.

[27] Steven M LaValle. *Virtual reality*. Cambridge university press, 2023.

[28] Antonio M Lopez et al. “Evaluation of methods for ridge and valley detection”. In: *IEEE Transactions on pattern analysis and machine intelligence* 21.4 (2002), pp. 327–335.

[29] Lee Markosian et al. “Real-time nonphotorealistic rendering”. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, pp. 415–420.

[30] Robert Patterson and Wayne L Martin. “Human stereopsis”. In: *Human factors* 34.6 (1992), pp. 669–692.

[31] Emil Praun et al. “Real-time hatching”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, p. 581.

[32] Whitman Richards. “Stereopsis and stereoblindness”. In: *Experimental brain research* 10.4 (1970), pp. 380–388.

[33] Szymon Rusinkiewicz. *trimesh2: A C++ Library for Triangle Meshes*. `https://gfx.cs.princeton.edu/proj/trimesh2/`.

[34] Szymon Rusinkiewicz et al. “Line drawings from 3D models”. In: *ACM SIGGRAPH 2008 classes*. 2008, pp. 1–356.

[35] Daniel Scherzer, Lei Yang, and Oliver Mattausch. “Exploiting temporal coherence in real-time rendering”. In: *ACM SIGGRAPH ASIA 2010 Courses*. 2010, pp. 1–26.

[36] Stephan Schnare and CapFrameX Team. *CapFrameX: Frame time capture and analysis tool based on Intel PresentMon*. Version 1.8.4. 2026. URL: `https://www.capframex.com/download`.

[37] Nicholas Sharp, Keenan Crane, et al. “GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing”. In: (2019).

[38] Mayank Singh and Scott Schaefer. “Suggestive Hatching.” In: *CAe*. 2010, pp. 25–32.

[39] Scott B Stevenson and Clifton M Schor. “Human stereo matching is not restricted to epipolar lines”. In: *Vision Research* 37.19 (1997), pp. 2717–2723.

[40] Frank Tong, Ming Meng, and Randolph Blake. “Neural bases of binocular rivalry”. In: *Trends in cognitive sciences* 10.11 (2006), pp. 502–511.

[41] Gang Xu and Zhengyou Zhang. *Epipolar geometry in stereo, motion and object recognition: a unified approach*. Vol. 6. Springer Science & Business Media, 2013.