Université de Montréal

Sketch-Based Interactive Shape Deformation using Shading Isophotes

par

Karl-Étienne Bolduc

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en Informatique

Orientation Imagerie

31 Décembre 2023

© Karl-Étienne Bolduc, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Sketch-Based Interactive Shape Deformation using Shading Isophotes

présenté par

Karl-Étienne Bolduc

a été évalué par un jury composé des personnes suivantes :

Noam Aigerman

(président-rapporteur)

Pierre Poulin

(directeur de recherche)

Mikhail Bessmeltsev

(codirecteur)

Bernhard Thomaszewski

(membre du jury)

Résumé

De plus en plus d'importance est accordée à la création d'objets 3D en raison des récents essors technologiques. Il est donc crucial de fournir des outils appropriés et accessibles aux utilisateurs de tous les horizons. Malheureusement, les outils traditionnellement utilisés en création 3D sont conçus pour des professionnels, exigent des formations complexes et de longue durée, et ne sont pas adaptés à ceux inexperimentés qui forment la vaste majorité des utilisateurs potentiels.

Nous proposons un outil de création simplifié qui utilise des méthodes inspirées d'esquisses. Dans un premier temps, le maillage désiré est créé à partir d'un contour tracé. L'intérieur est gonflé suivant la méthode de Dvoroznak et al. [**10**]. Dans un deuxième temps, la hauteur des sommets du maillage est manipulée en modifiant les courbes formées par l'ombrage. Cet ombrage provient d'un modèle de réflexion Lambertien pour une lumière directionnelle donnée.

Notre méthode consiste à utiliser les courbes formées par la méthode des charactéristiques associée au problème de figure dérivée de l'ombre (*Shape-From-Shading*). Avec les courbes, nous identifions les régions affectées par la modification de l'ombrage. L'une de ces régions sera utilisée pour interpoler l'ombrage d'après la nouvelle isophote. À partir de ce nouvel ombrage, les courbes de la méthode des characteristiques seront utilisées afin de trouver le nouveau déplacement en s'assurant d'altérer uniquement la région affectée par le changement dans l'ombrage. Les maillages créés peuvent ensuite être combinés suivant la méthode proposée par Dvoroznak et al. [**10**] afin de former un maillage unique et complexe. Notre outil se veut plus intuitif que les outils traditionnels de création. Nos résultats en illustrent le potentiel.

mots-clés : Outils de création, figure dérivée de l'ombre, Ombrage inverse, Réflectance Lambertienne

Abstract

Due to recent technological advances, the creation of 3D objects is becoming more important. It is critical to offer appropriate and accessible tools to users from diverse backgrounds. Unfortunately, the tools traditionally used in 3D creation are designed for professionals, require complex and time-consuming training, and are unsuitable for inexperienced users who form the vast majority of potential users.

We propose a simplified creation tool that uses sketch-based methods. First, the desired mesh is created from a traced outline. The interior is inflated following the method of Dvoroznak et al. [**10**]. Second, the height (displacement) of the mesh is achieved by altering the strips created by shading. Shading is the result of a Lambertian reflection model for a given directional light.

Our method consists of using the strips from the method of characteristics applied to solve Shape-From-Shading. Using the strips, we identify the regions affected by the change in shading. One of these regions will be used to interpolate the shading according to the new isophote. From this new shading, the characteristic strips will be used to find the new height, ensuring that only the region affected by the change in shading is altered. The meshes created can then be combined, inspired by the method proposed by Dvoroznak et al. [**10**] to form a single, complex mesh. Our tool is designed to be more intuitive than the ones provided by professional 3D software. Our results illustrate its potential.

keywords: Creation Tool, Shape-From-Shading, Inverse Shading, Lambertian Reflectance

Contents

Résumé v

Abstract vii

List of terms and abbreviations xi

Acknowledgements xiii

Chapter 1. Introduction 1

Chapter 2. Literature Review 5

2.1. Geometry Processing 5

2.2. Rendering 10

2.3. Inverse Rendering 16

2.3.1. Classic Shape-From-Shading 16

2.3.2. Shape-From-Shading Related Problems 23

2.4. 3D Creation 25

Chapter 3. Background 31

3.1. Method of characteristics for SFS 31

3.2. *Monster Mash* Approach 35

Chapter 4. Methodology 41

4.1. System Overview 42

4.2. Inflation 45

4.3. Bounded Band and Impacted Region 47

4.4. Localized Shading Interpolation 50

4.5. Strip Integration 51

4.6. Shape Reconstruction 52

4.7. Assembling Layers 53

Chapter 5. Results 57

Chapter 6. Conclusion 67

References 71

List of terms and abbreviations

2D	Two Dimensional
3D	Three Dimensional
BRDF	Bidirectional Reflectance Distribution Function
PDE	Partial Differential Equation
RBF	Radial Basis Function
SFS	Shape-From-Shading

Acknowledgements

I am deeply grateful for the support and guidance I received throughout this project. I want to express my appreciation to my partner for her unwavering support, love, and understanding during this journey. I also owe special thanks to my mother for her belief in me. Special thanks to my advisor Pierre Poulin and co-advisor Mikhail Bessmeltsev for their invaluable advice and multiple revisions. I am grateful for the valuable feedback from my committee that helped improve this thesis.

I am thankful for my experience at Ubisoft La Forge, where I had the opportunity to meet Damien, Jean-Philippe, and Arnaud. The willingness to share their passion was inspiring. Additionally, I am grateful to Andrea for her involvement and for making me feel an integral part of the team at La Forge. Likewise, I deeply value the enriching discussions with my colleagues at LIGUM - Elsa, Cédric, Ivan, Caio, Mathis, Bowen, Mathieu, and William.

Thanks to my family, friends, and all the participants for their understanding and support.

I also wish to express my appreciation to NSERC for their financial support. This includes the NSERC Discovery grant facilitated by Pierre and the NSERC Nova grant, obtained through Mikhail and Pierre.

Chapter 1

Introduction

Michelangelo is probably one of the best known artists for his sculptures such as *David* (Figure 1.1) and his frescoes on the ceiling of the Sistine Chapel. These creations were born from the technological developments and knowledge of their time. Fast-forward to the era of the computer revolution, the blend of art and technology finds a parallel in the field

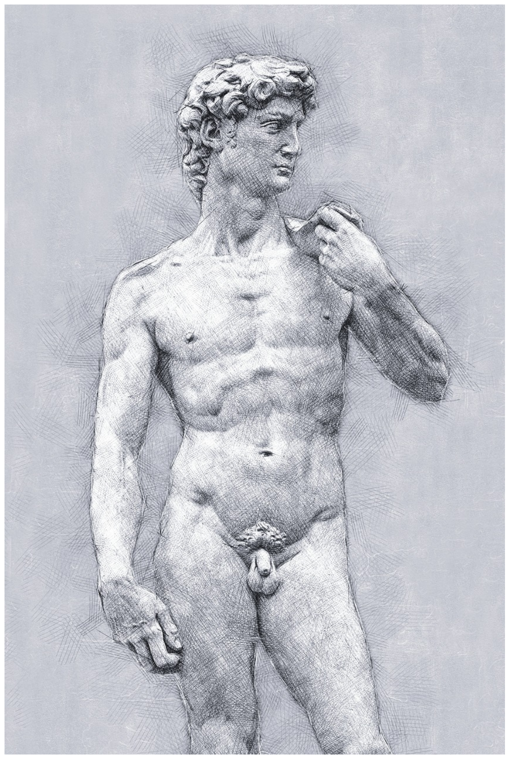


Figure 1.1 – Sketching of the statue of *David* made by Michelangelo. Sketch by azazelok from `https://pixabay.com/photos/michelangelo-david-revival-art-2739280/`.

of computer graphics for synthetic images. Image synthesis frequently involves a virtual scene with geometric shapes, materials and textures, virtual light sources, virtual cameras, and other elements such as light transport algorithms. Tools for these creations, found for instance in software like Blender and Maya, are extensively utilized in diverse artistic media, including films, video games, and documentaries.

In contrast, many individuals possess an ability for drawing, a skill that is inherently more intuitive yet paradoxically underutilized in computer graphics due to hidden complexities. A key element in translating two-dimensional (2D) drawings to three-dimensional (3D) shapes lies in our ability to perceive depth from shades of gray[1] in an image. The interaction between light and surface properties, such as orientation and materials, create gradients of gray that the human visual system intuitively understands, influencing depth perception. This gradient results from a combination of shape curvature, texture, shadows, occlusion, attenuation, perspective, etc., which makes this problem heavily underconstrained. Despite its naturalness, deducing 3D elements from 2D information remains a complex task. While depth perception has been a subject of numerous studies, it remains an area with much to be understood. It is still unclear if we can consistently and accurately retrieve a shape, even if our visual system suggests it may. This limitation is exploited in several optical illusions.

In our research, we seek to use the gradient of gray to manipulate geometric features, giving users the ability to naturally and intuitively modify geometric elements of an object while reducing the learning curve for creation tools. It expands on *Monster Mash* from Dvoroznak et al. [**10**], a tool to create shapes of more organic blobby-like objects by drawing their contours. In a simplified setting with purely Lambertian surfaces and control for an incident light direction, we seek to reuse developments made since the 1970s on Shape-From-Shading problems [**20, 19, 14, 44, 59, 28, 11, 65, 46, 22**] and real-time shading in order to manipulate shapes. The resulting shapes are displayed in real time for direct feedback in an iterative editing process (Figure 1.2).

This thesis is organized as follows. We begin with a literature review in Chapter 2 starting from geometry processing, rendering, and inverse rendering, to follow with 3D creation tools. Then, we elaborate more on the background needed in Chapter 3 to implement our method. In Chapter 4, we begin with a short overview of our system. Afterwards, we describe the ins and outs of our method and the underlying concepts. In Chapter 5, we present our results

1. On a computer, colors are represented as additive color channels RGB. Gray appears when the intensity of each channel has the same value.

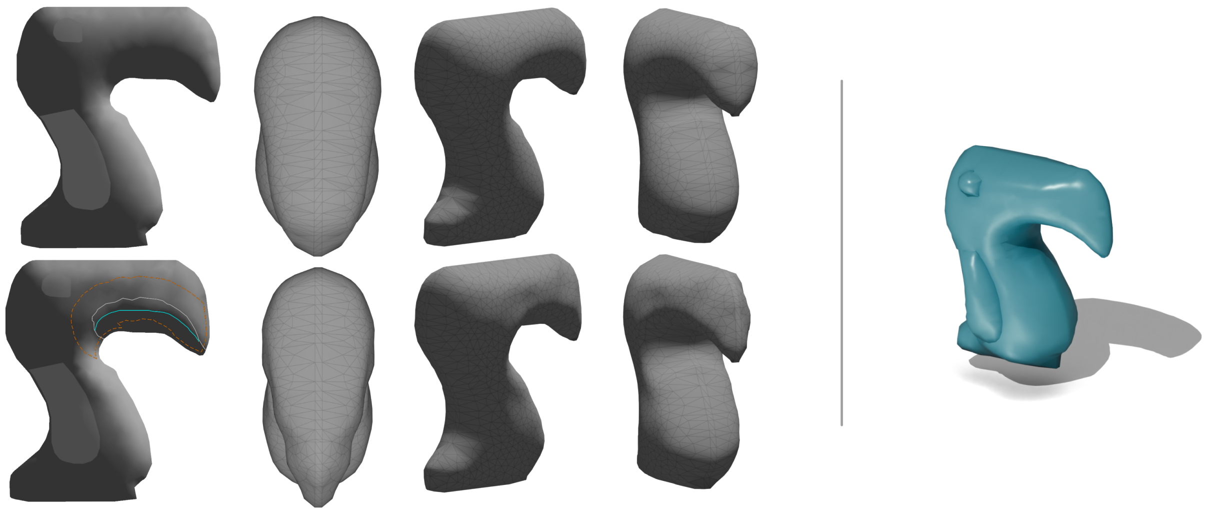


Figure 1.2 – On the left, the leftmost column shows the shading of the shape before (top) and after (bottom) it has been edited with our method. The three other columns show the two meshes under different viewpoints. On the right, we display the final mesh after several iterations.

with an in-depth discussion of different cases. Finally, we conclude our work with a brief discussion on future improvements in Chapter 6.

Chapter 2

Literature Review

In this chapter, we begin with an overview of the fundamental principles in geometry processing. We explain some elements of radiometry and how rendering is done. Then, we provide an overview of inverse rendering techniques to retrieve three-dimensional (3D) features from shading. Finally, we present artistic design tools that use concepts from geometry processing, rendering, and inverse rendering.

2.1. Geometry Processing

Geometry processing is a specialized area of study encompassing the manipulation and analysis of geometric structures, particularly in the realm of computer graphics and computational mathematics. In its simpler form, geometry processing is an extension of signal processing that regroups operations on shapes.

Geometry processing is divided into three stages: acquisition, processing, and consumption. In the past decades, acquisition has been more accessible with the development of new technologies like LiDAR, ultrasound, 3D scanners, and so on, and the advancement in modeling tools. The resulting geometric data are processed through reconstruction, filtering, re-meshing, and parameterization algorithms. Consumption refers to the final stage where the result is utilized by another program, a computer display, a 3D printer, etc. Geometry processing algorithms often reuse calculus and differential geometry concepts.

The concept of shape is central in geometry processing. Mathematically, shapes are characterized by geometric and topological properties. Geometric aspects focus on extrinsic attributes like position, normal, curvature, etc. Topological properties, on the other hand,

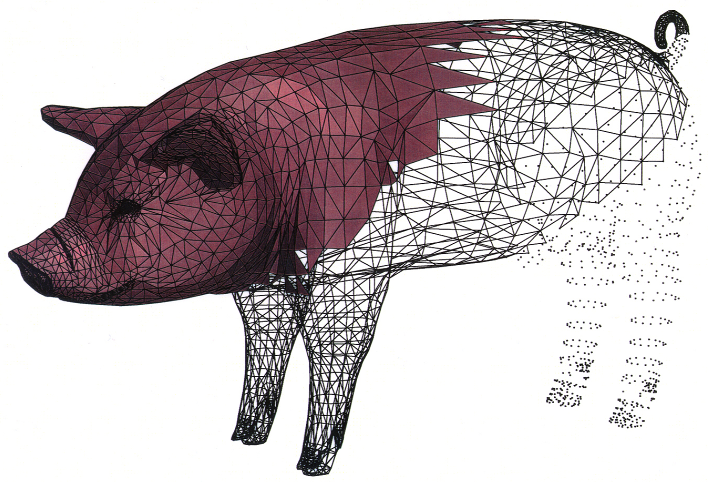


Figure 2.1 – Example of a triangle mesh. In this case, we can see a mesh where the points are vertices, dark segments are the edges of triangles, and colored regions are the triangles with shading. Image from `http://web.mit.edu/manoli/crust/www/slides/piggy.jpg`.

delve into the intrinsic characteristics of a shape that remain unchanged despite deformations, such as bending or stretching. Notable examples are properties like orientability and connectivity.

Shapes can be represented in two primary ways: through their boundaries or their volumes. Our work primarily focuses on the former, specifically on surfaces. Surfaces in mathematics are structures that are embedded, at least, in three-dimensional space $\mathbb{R}^3$. These are not just any random assemblages in space; rather, they are defined by properties like smoothness and orientability, properties that must hold across every small region of the surface, thus characterizing the entirety of it.

The representation of surfaces is broadly categorized into three types: implicit, parametric, and explicit. Implicit representations define shapes through functions where each parameter cannot be solved directly, such as Signed Distance Functions (SDFs). An SDF assigns a value to each point in space, indicating its "distance" from a surface. Parametric surfaces are surfaces where each parameter depends on other common parameters. They can be Bézier surfaces, bicubic polynomial patches, spline-based surfaces, etc. Explicit surfaces

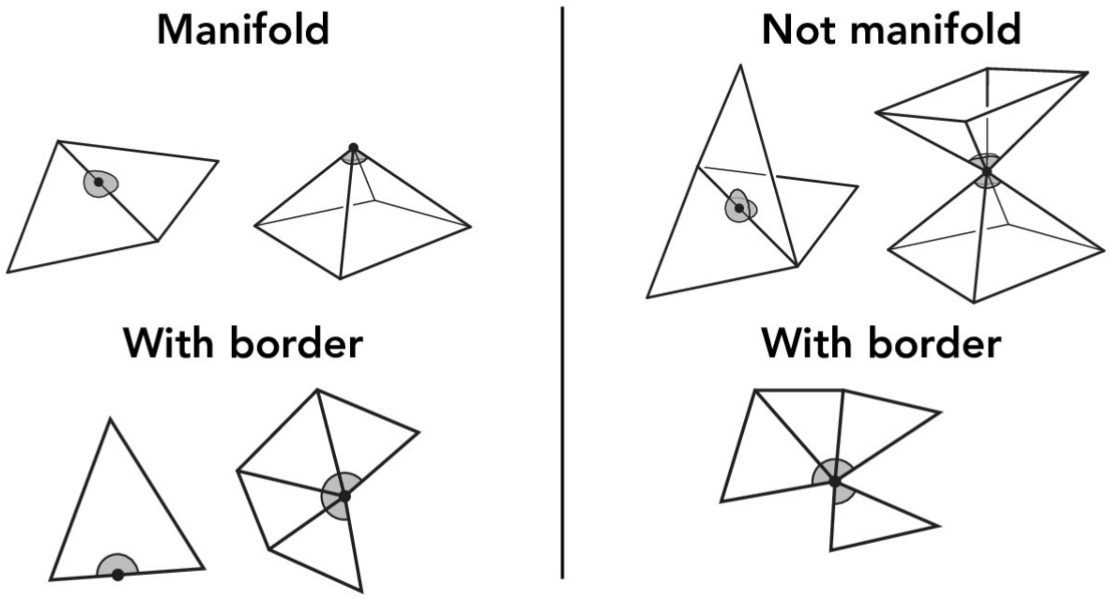


Figure 2.2 – Examples of different types of manifold and non-manifold structures for triangle meshes. A manifold ensures that one can draw a continuous, closed contour around a vertex, passing through all connecting edges without any discontinuity so that it forms a half-disk-like shape. Image adapted from `https://cs184.eecs.berkeley.edu/sp19/lecture/8-17/meshes-and-geometry-processing`.

are functions where one of the parameters can be solved directly based on the others. In computer graphics, explicit surfaces are, most of the time, represented through a finite set of elements. They can be point clouds, graphs, triangle meshes, etc. Explicit surfaces are more widespread than other representations.

Our work primarily focuses on triangle meshes, a specific type of graph (Figure 2.1). These meshes consist of a collection of vertices and triangles (V, F), arranged under particular connectivity rules such that the region around each vertex forms a manifold. A manifold (Figure 2.2) ensures that one can draw a continuous, closed contour around a vertex, passing through all connecting edges without any discontinuity so that it forms a half-disk-like shape.

As with signal processing, geometry processing often requires the use of partial differential equations (PDEs) for different applications. A PDE can be described as an operator $\mathcal{O}$ acting over a function f such as ∇f, $f\Delta f$, and so on.

A typical PDE over a smooth surface Ω can have the following form

$$\mathcal{O}(f(x)) = g(x) \quad x \in \Omega. \tag{2.1.1}$$

Common boundary conditions, for $x \in \partial\Omega$, are Dirichlet $f(x) = h(x)$, Neumann $\frac{df}{dx}(x) \cdot \mathbf{n} = h(x)$, or Robin $af(x) + b\frac{df}{dx}(x) = h(x)$. Boundary conditions are necessary to have a well-posed PDE. As stated by Hadamard [**17**], a PDE is well-posed if a solution exists, is unique, and the solution depends continuously on the initial or boundary conditions.

However, PDEs cannot be used directly on a triangle mesh because the domain of a mesh is non-trivial, discrete, and embedded in $\mathbb{R}^3$. Equation 2.1.1 is often referred to as the strong form because f needs to be well-defined at every point and differentiable up to the degree of the PDE. It means that we cannot use it directly on a discrete set of triangles. Instead, using the fundamental lemma of variational calculus, we can rewrite it in its weak form

$$\int_\Omega (\mathcal{O}(f(x)) - g(x))v(x)dx = 0 \quad \forall v(x) \in \mathcal{S} \tag{2.1.2}$$

where $\mathcal{S}$ is a set of test functions where $v(x)$ is assumed to be smooth.[1] The weak solution lets us solve the PDE over a region even if it is not well-defined.

In geometry processing, most operators are typically linear operators $\mathcal{O}_{\text{lin}}$, which means that they need to respect $\mathcal{O}_{\text{lin}}(u + v) = \mathcal{O}_{\text{lin}}(u) + \mathcal{O}_{\text{lin}}(v)$ and $\mathcal{O}_{\text{lin}}(c \cdot u) = c \cdot \mathcal{O}_{\text{lin}}(u)$. There exist plenty of operators for intrinsic or extrinsic linear operators [**62**]. We seek to discretize the Laplacian PDEs so that we can express them on a mesh

$$\begin{cases} \Delta f(x) = 0 & x \in \Omega \\ f(x) = h(x) & x \in \partial\Omega \end{cases} \tag{2.1.3}$$

where Ω is the domain of integration, and $\partial\Omega$ is the boundary. The second equation of the system corresponds to the boundary condition.

We can rewrite the system in its weak form

$$\begin{cases} \int_\Omega (\Delta f(x) - g(x))v_j(x)dx = 0 \\ f(x) = h(x) \end{cases} \tag{2.1.4}$$

where $v_j \in \mathcal{S}$.

By using the Green identity, Equation 2.1.4 becomes

$$\begin{cases} -\int_\Omega \nabla v_j(x) \cdot \nabla f(x)dx + \int_{\partial\Omega} v_j(x)(\mathbf{n} \cdot \nabla f(x))dx - \int_\Omega v_j(x)g(x)dx = 0 \\ f(x) = h(x). \end{cases} \tag{2.1.5}$$

1. The smoothness criteria can often be relaxed for lower-order differentiable functions.

If the mesh does respect the manifold property and is completely closed, we can discard the second term because the domain has no boundary. It is important to note that **n** for an embedded surface refers to the normal along the boundary in the surface (i.e., tangent to the surface) and not the normal at the surface (i.e., perpendicular to the surface). If it does not, it is common to assume that the test function vanishes closer to the boundary [2].

$$\begin{cases} -\int_\Omega \nabla v_j(x) \cdot \nabla f(x)dx - \int_\Omega v_j(x)g(x)dx = 0 \\ f(x) = h(x). \end{cases} \tag{2.1.6}$$

We can express $f(x) = \sum_i f_i v_i(x)$, $g(x) = \sum_i g_i v_i(x)$, and $h(x) = \sum_i h_i v_i(x)$ using a basis $v(x)$. This approach is called the Galerkin finite element method where we assume that the basis of $f(x)$, $g(x)$, and $h(x)$ are the same functions part of the test set.

$$\begin{cases} \sum_{i \in |\mathcal{S}|} \int_\Omega -f_i \nabla v_i(x) \cdot \nabla v_j(x)dx - \int_\Omega g_i v_i(x) v_j(x)dx = 0 \\ f_k = h_k \qquad \qquad \forall k \in |\partial\Omega|. \end{cases} \tag{2.1.7}$$

If such a test set exists, we can rewrite Equation 2.1.7 into a linear system [3]. The conventional test function is to use the hat function, which is one at a vertex position and zero otherwise, where values in-between vertices are piecewise linearly interpolated (Figure 2.3).

Then, we have

$$\begin{cases} \mathbf{Lf} = \mathbf{Mg} \\ f_k = h_k \qquad \forall k \in |\partial\Omega| \end{cases} \tag{2.1.8}$$

where each row of **f** and **g** represents the coefficients for each test set v_i.

Without going into detail, Pinall et al. [**47**] show how to compute **L** using trigonometry and refer to it as the *cotangent Laplacian.* **M** is referred to as the *mass matrix* and can be computed using the barycentric or Voronoi area around each vertex [**36**]. The book of Bosh et al. [**5**] provides additional information.

2. Boundary is often used to represent two similar concepts and can be quite confusing. The boundary can be referred to as the actual border of a domain like the border of a mesh or the condition that we impose on a domain of integration which could be inside a mesh. The two are often mismatched because they are often the same.

3. Even though we specify that the test function at the boundary vanishes, we can make a stronger claim that every value at the boundary should be equal, even if the test function at the boundary vanishes.

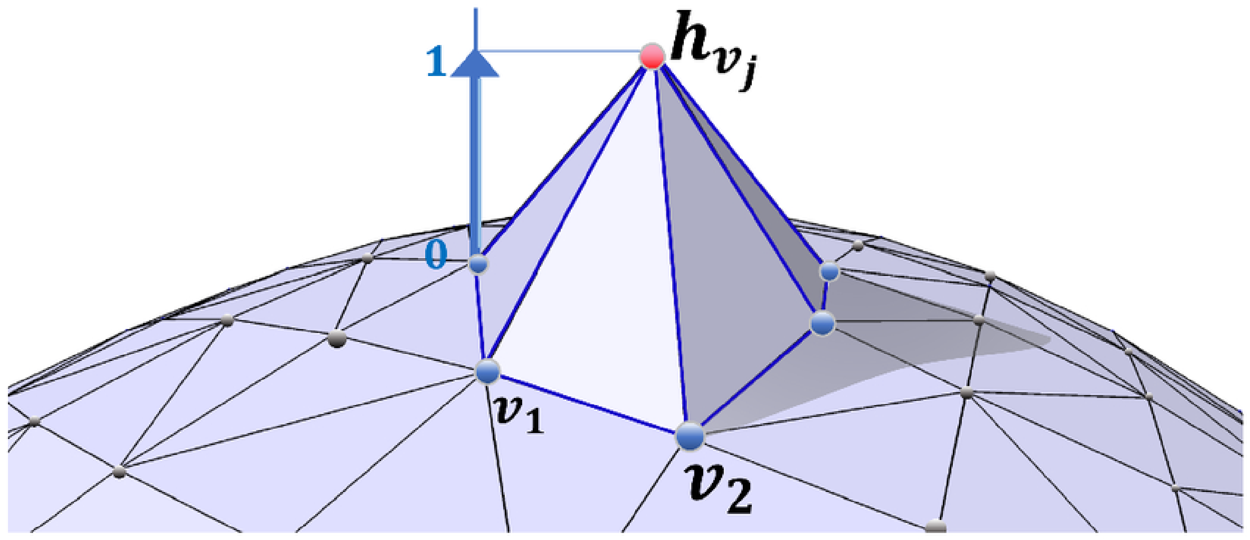


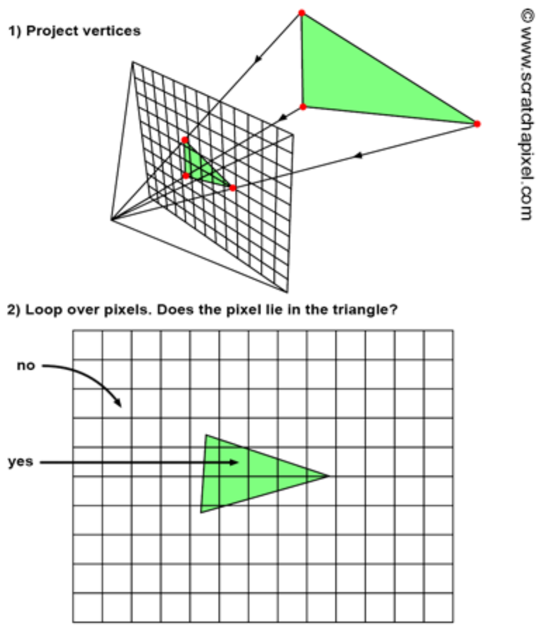
Figure 2.3 – Hat function defined over a triangle mesh. Image from `https://www.researchgate.net/figure/Illustration-of-a-hat-function-51-defined-over-a-2D-simplicial-complex-embedded-in_fig9_352176002`.

Other methods like boundary element methods and discrete exterior calculus can make it easier to discretize our PDE, but those methods are out of the scope of this literature review. Unfortunately, no perfect solutions exist and they all have some kind of trade-off. The Galerkin approach is more common in practice.

2.2. Rendering

Rendering in computer graphics is the process of generating images that visually represent a scene, composed of elements like geometries, light sources, and materials, from a specific viewpoint, or camera. It involves simulating how light interacts with surfaces, as a combination of geometry and material properties.

In current applications, the majority of images are rendered in 2D and displayed on flat computer screens. These images are essentially a grid composed of numerous picture elements, commonly known as pixels, which are arranged in rows and columns. Each pixel contains a number of channels that carry color information. The most widely used format for this purpose is the RGB color model. RGB, an acronym for Red, Green, and Blue, is an additive color model where various colors are produced by blending different intensities for each color channel.



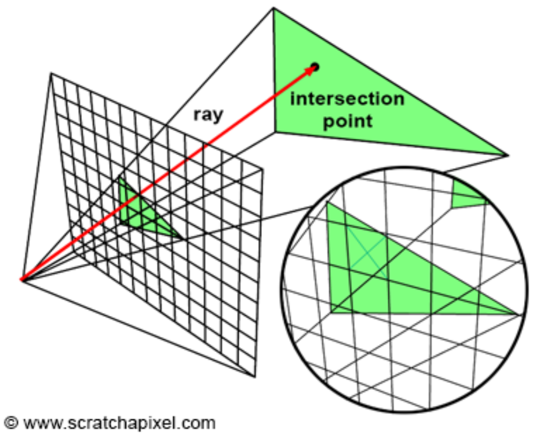


Figure 2.4 – Two common integration processes to display a scene into a 2D image: rasterization (left) and ray tracing (right). Figures from `https://www.scratchapixel.com/`.

Rendering stands as one of the earliest and continually evolving topics within the domain of computer graphics. It primarily aims to replicate the visual impression of a scene, encapsulating aspects like lighting, shading, materials, and other visual characteristics to craft either realistic or stylized representations. The rendering process typically follows a structured approach known as the graphics pipeline, which converts scene information into a format optimized for hardware acceleration, notably on Graphics Processing Units (GPUs). GPUs are specially architected to execute highly parallel programs, commonly referred to as shaders for graphics-related tasks and beyond. Then, the data is integrated to form a final image. The integration can come from various rendering methods, but rasterization and ray tracing have emerged as the most prevalent methods. They have become feasible thanks to significant technological and algorithmic advancements over recent decades.

Rasterization, as depicted in Figure 2.4 (left), is a process that transforms vector-based information into a discretized image. This technique begins by taking geometric primitives from the scene consisting of polygons, lines, or points. These primitives are transformed into the frustum of the camera—a truncated pyramidal shape defining the directly visible region of 3D space from the camera's perspective. In the frustum, two axes correspond to the screen coordinates, while the third axis represents "depth" relative to the camera.

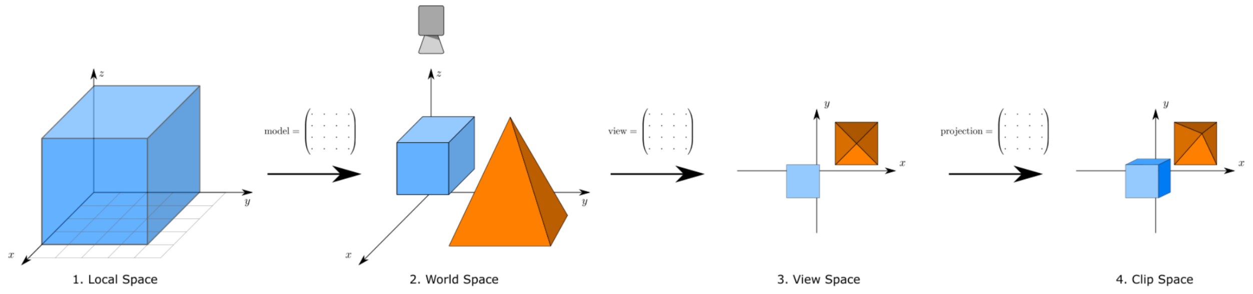


Figure 2.5 – Series of transformations referred to as Model View Projection Transformation. Image taken from Kai Lawonn Lecture in Computer Graphics at `https://vis.uni-jena.de/?page_id=540`.

The depth is crucial as it determines the correct order in which primitives are viewed. Content outside the frustum is not considered for rendering. The rasterization process continues with a scan-conversion algorithm to determine which pixels of the image fall inside or on the boundary of a primitive and should thus be filled. During this phase, attributes assigned to the vertices of primitives, such as color normal, or texture coordinates, are interpolated across the covered pixels. This interpolation is key to producing efficiently more complex visual effects, simulating light interactions within the rendered image.

In the context of rasterization, to minimize data manipulation on the GPU and under the assumption that geometric shapes remain relatively unchanged most of the time, affine transformations are utilized to convert points between coordinate spaces. The process typically comprises several stages. Initially, vertices are transformed from a local space to a global space. These transformed vertices are then adjusted to the camera's viewpoint. Finally, the vertices are projected onto a 2D screen using either an orthographic or perspective projection. This series of transformations is often referred to as the MVP (Model-View-Projection) model (Figure 2.5).

Ray tracing, illustrated in Figure 2.4 (right), employs a distinct approach compared to rasterization. This technique involves tracing the path of individual rays that originate from the camera aperture and pass through each pixel on the image plane. These rays interact with the geometry of the scene. The color of each pixel is determined based on these interactions, considering the intersections of rays with scene objects, and accounting for light and material properties of the objects. While basic ray tracing provides a fundamental framework, more advanced implementations exist to accurately render complex surfaces.

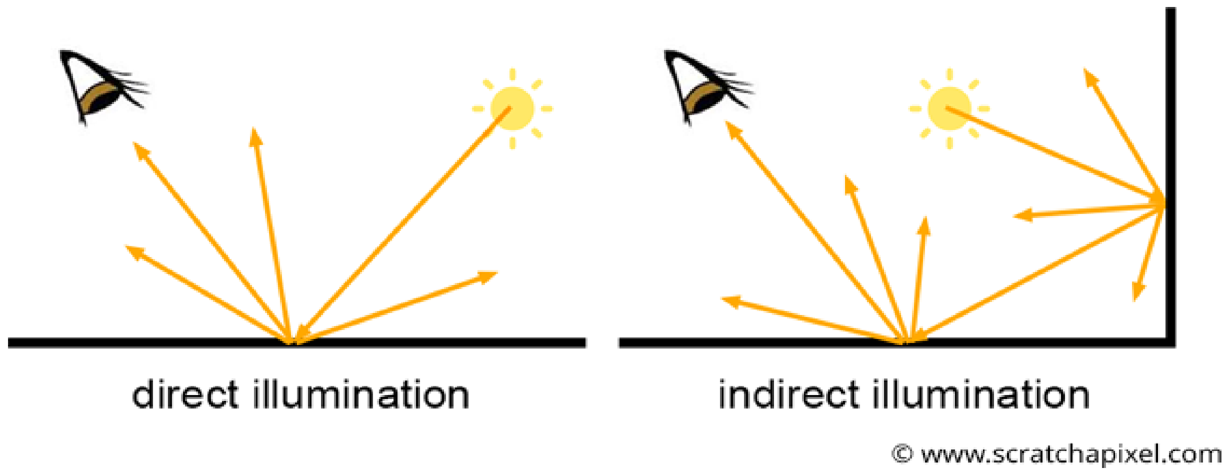


Figure 2.6 – Direct (left) and indirect (right) illuminations from a light source. Direct illumination is caused by light bouncing once off the surface and reaching the eye directly. Indirect illumination is when light bounces on multiple surfaces before going to the eye. Figures from `https://www.scratchapixel.com/`.

These include surfaces that are reflective, refractive, or composed of microfacets. However, such advanced techniques are beyond the scope of our work.

Illumination is elementary to create stunning visual effects (Figure 2.6). Illumination, in the context of 3D scenes, pertains to the realistic rendering of lighting effects, accounting for how light rays interact with various surfaces. This aspect is fundamental in creating computer-generated images that are both lifelike and visually compelling. Often referred to as global illumination, it encompasses two primary components. Direct illumination involves light emanating from a light source and directly reaching and reflecting off surfaces, before reaching the camera. It contributes to the most immediately noticeable lighting effect. Indirect illumination is more complex, as it involves light rays that bounce off multiple surfaces before ultimately being captured by the camera. Surface regions that do not interact directly with light create regions that we refer to as shadows. Those shadows are cast by the other geometric elements in the scene and are referred to in the computer graphics community as shadow casting. These elements together create a more dynamic and realistic portrayal of lighting in 3D environments.

To model complex illumination, the theory of radiometry has proven to be instrumental. It enables the realistic rendering of light and its interactions within digital environments.

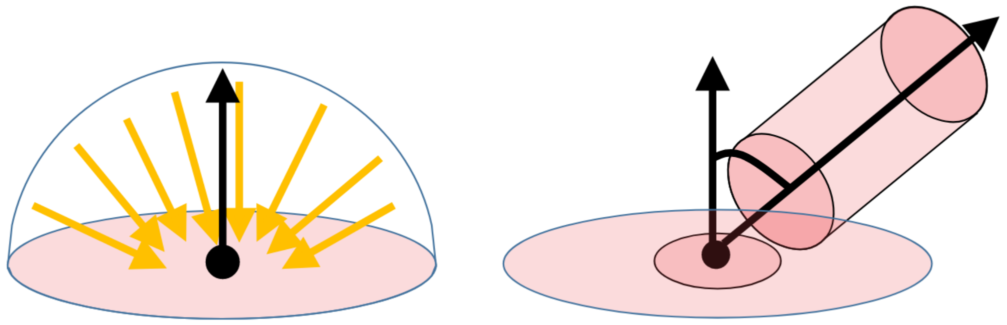


Figure 2.7 – Two concepts from radiometry: irradiance (left) and radiance (right). Irradiance is the total amount of light that is flowing from or into a point. Radiance is the amount of light received at a point from one direction. Figures provided by Adrien Gruson and adapted.

Radiometry focuses on measuring light propagation. Within this field, two central concepts are often discussed in rendering: irradiance and radiance. Irradiance quantifies the amount of light [4] received by a point on a surface from all directions above it. In contrast, radiance measures the amount of light passing from a point on a surface in a particular direction. These concepts are essential for understanding how light interacts with surfaces (Figure 2.7).

Based on radiometry and optical geometry, Kajiya introduced the rendering equation [**24**] in 1986 as

$$\begin{aligned} L_o(\mathbf{x}, \omega_o) &= L_e(\mathbf{x}, \omega_o) + L_r(\mathbf{x}, \omega_o) \\ L_r(\mathbf{x}, \omega_o) &= \int_{H^2} f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \hat{\mathbf{n}}) d\omega_i \end{aligned} \tag{2.2.1}$$

with the following terms:

L_o is the total radiance

L_e is the emitted radiance

L_r is the reflected radiance

L_i is the incident radiance

$\mathbf{x}$ is a point in space

$\hat{\mathbf{n}}$ is the normal at the location in space

ω_o is the direction of the outgoing light

4. Quantity of light is often referred to as flux in the literature.

ω_i is the direction of the incident light

H^2 is a subspace that represents all possible directions in a hemisphere

f_r is the Bidirectionnal Reflectectance Distribution Function (BRDF).

Multiple work have reused Equations 2.2.1. The rendering equation, in itself, is very powerful because it can model a very large variety of phenomena.

At the core of Physics-based rendering (PBR) lies the Bidirectionnal Reflectectance Distribution Function (BRDF) f_r, a critical component that defines how light scatters off a surface in different directions, enabling the simulation of complex phenomena such as reflections, refractions, and subsurface scatterings. PBR emphasizes the realistic simulation of multiple light interactions with surfaces and materials. This approach is favored for its ability to produce highly realistic and physically accurate visual results.

A simple BRDF commonly used is the Lambertian model, which assumes that light is uniformly redistributed in intensity in the hemisphere above a point

$$f_r(\omega_i, \omega_o) = \frac{\rho}{\pi} \tag{2.2.2}$$

where ρ is referred to as the albedo.

If we suppose that we have a light source in one direction where $L_i(\mathbf{x}, \omega_i) = \delta(\omega_i - \mathbf{l})$ and light direction $\mathbf{l} \in H^2$, by using the rendering equation 2.2.1, we obtain a well-known formulation

$$\begin{aligned} L_o(\mathbf{x}, \omega_o) &= \int_{H^2} f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_\mathbf{i})(\omega_i \cdot \hat{\mathbf{n}}) d\omega_i && (2.2.3) \\ &= \int_{H^2} \frac{\rho}{\pi} \delta(\omega_i - \mathbf{l})(\omega_i \cdot \hat{\mathbf{n}}) d\omega_i && (2.2.4) \\ &= \frac{\rho}{\pi}(\hat{\mathbf{n}} \cdot \mathbf{l}). && (2.2.5) \end{aligned}$$

If we suppose that $\rho = \pi$ and that the integration is done over the whole sphere S^2 of directions, we can rewrite the equation. The part of the domain of integration that is not part of H^2 should not affect the result because no light can traverse below the surface[5]. This part of the domain is responsible for darker regions observed in a shaded image. The contribution of Equation 2.2.5 in this region is none.

$$R(\hat{\mathbf{n}}) = \max(\mathbf{l} \cdot \hat{\mathbf{n}}, 0), \tag{2.2.6}$$

5. We assume that the surface is opaque and no light cannot traverse it in any way.

Equation 2.2.6 is known as the Lambertian reflectance. More complex BRDFs exist such as from Phong-Blinn [**4**], Ward [**63**], Oren-Nayar [**42**] and Matusik's MERL measurement database [**35**]. When the BRDF varies over a surface, it is qualified as an svBRDF, and as tsvBRDF when it also varies over time. However, modeling more complex light behaviors through BRDFs is not within the scope of this thesis. The variation of reflectance observed at each point corresponds to what we commonly refer to as shading. The normal where the reflectance is zero is commonly referred to as self-shadowing and is responsible for the dark region observed in a shaded image.

2.3. Inverse Rendering

Inverse rendering is an area of study that concentrates on retrieving all kinds of physical properties in a scene from single or multiple images. The problem of inverse rendering is one of the fundamental problems of computer vision. Computer vision focuses on acquiring, and analyzing images to mimic the intricacy of human perception of color and shape. It enables machines to interpret visual data from the real world, extracting high-dimensional data from images captured in various environments. This process goes beyond mere recognition and categorization of visual elements; it involves understanding spatial and temporal relationships, identifying patterns, and interpreting different environmental contexts. Shape-From-Shading (SFS) was one of the first problems introduced to retrieve the displacement of a shape from its shading in computer vision.

2.3.1. Classic Shape-From-Shading

From the 1970s to the 2000s, Shape-from-Shading has been an area of intense research focus. These methods were designed to deduce 3D shape information from a single image. By analyzing the shading within the image, SFS methods estimate the depth of surfaces at each point. This process is based on the assumption that lighting conditions and reflectance properties of surfaces are uniform and either known or could be estimated.

Depth perception is heavily influenced by shading. Barrow et al. [**2**] highlight that shading is the result of a combination of illumination, reflectance, and surface orientation information. They note that this combination is not unique, indicating the complexity of deciphering shading and depth. Ramachandran [**51**] discovered several common assumptions about shading. He found that people typically presume incident light to originate from above, illumination to be uniform and consistent, and perceived shape to be influenced by boundaries (Figure 2.8).

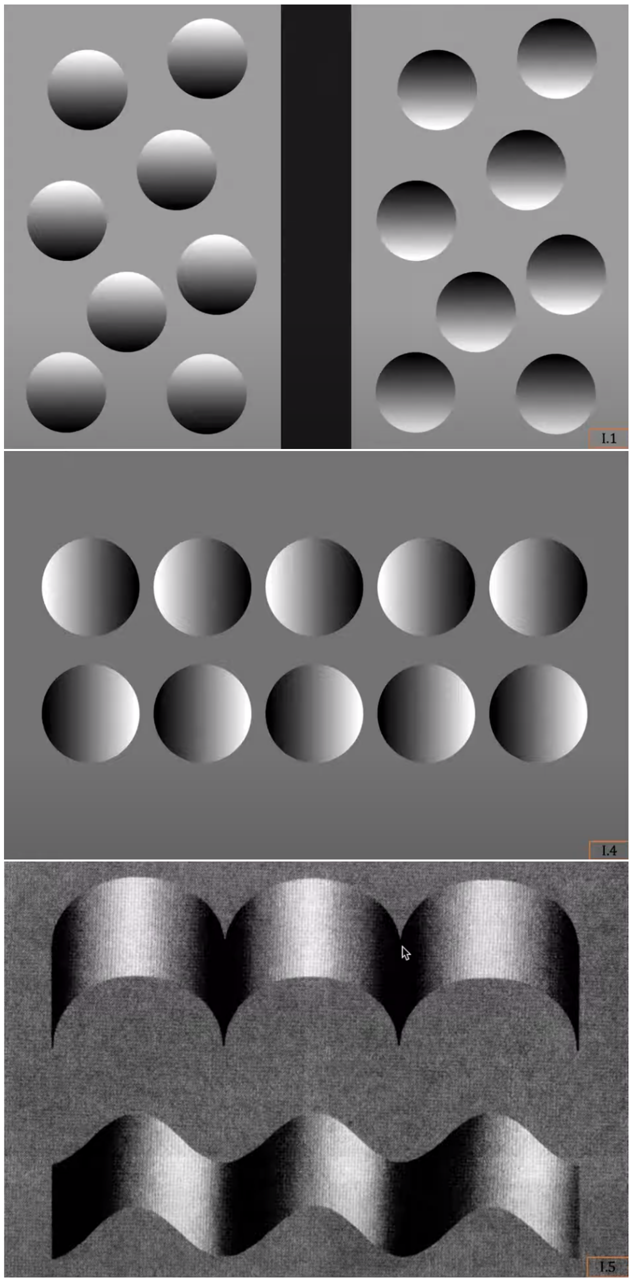


Figure 2.8 – Optical illusions made by our perception from shading, as studied by Ramachandran [**51**]. Image taken from `https://fpcv.cs.columbia.edu/`.

However, he also observed that these global assumptions about lighting and shading can be altered by an individual's prior knowledge or experience. This suggests that our understanding and interpretation of shading, and hence depth, are not only influenced by the immediate visual cues but also by our cognitive process and prior experiences.

Typically, in SFS, lighting conditions are assumed to be known as well as material properties such as the BRDF for a surface. The process is to extract information from the irradiance and to retrieve the original shape. In its simplistic form, the light reflected is assumed to follow a Lambertian distribution. Instead of Equation 2.2.6, reflectance R is reformulated for an implicit surface $z(x,y)$ with the coordinates $p = \frac{dz}{dx}$ and $q = \frac{dz}{dy}$. The normal of such an implicit surface is colinear with

$$\begin{aligned}\mathbf{n} &= \begin{bmatrix}-\frac{dz}{dx} & -\frac{dz}{dy} & 1\end{bmatrix}^T \\ &= \begin{bmatrix}-p & -q & 1\end{bmatrix}^T. \end{aligned} \tag{2.3.1}$$

The inverse transformation is

$$\begin{aligned} p &= -\frac{n_x}{n_z} \\ q &= -\frac{n_y}{n_z}. \end{aligned} \tag{2.3.2}$$

Reflectance R can be written as

$$R(p,q) = \frac{-l_x p - l_y q + l_z}{\sqrt{(1 + p^2 + q^2)}} \tag{2.3.3}$$

where l_x, l_y, and l_z define the normalized direction of light. Each level-set of the Lambertian reflectance (Equation 2.3.3) is an isophote. An isophote corresponds to a curve of equal brightness in the observed image. A complex surface can thus have disconnected isophotes. A null brightness occurs in regions in full shadow, and an associated isophote is ill-defined. However, right on the border of such a region, we will consider the isophote of null brightness for our manipulation.

Even if other more advanced BRDFs could be used for SDF, the problem seems to be already difficult enough to solve because of the inherent ambiguity related to shading.

The SFS problem, in the computer vision literature, is then generally acknowledged to be

$$E(x,y) = R(p,q) \tag{2.3.4}$$

where $E(x,y)$ represents the shading intensity observed at a given point in the image plane xy.

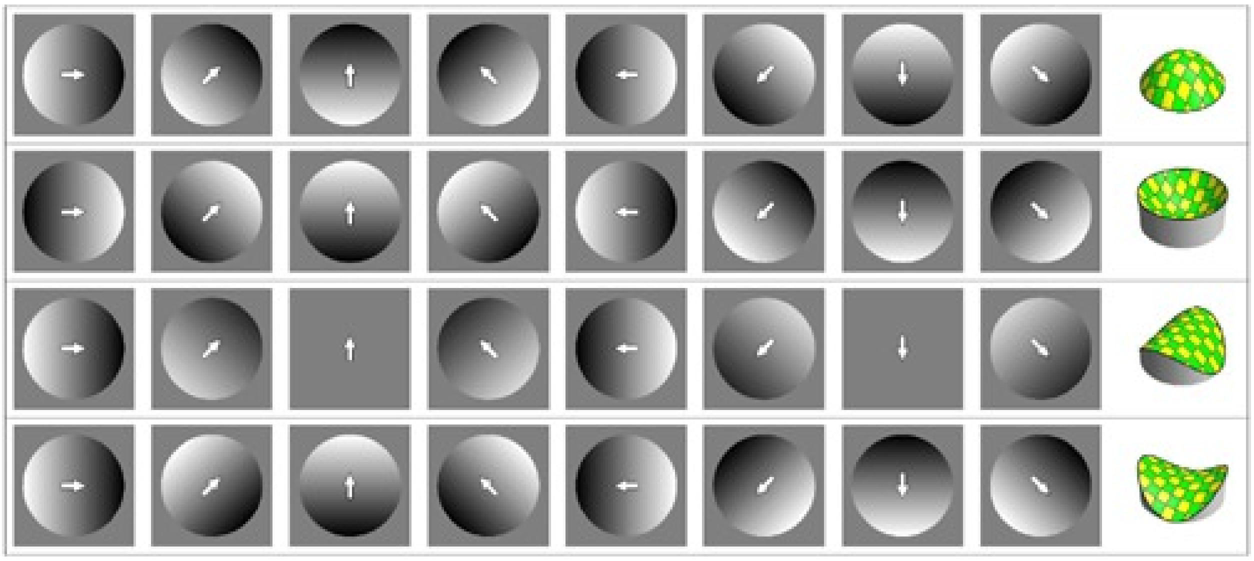


Figure 2.9 – Shading ambiguity even for a simple surface is one of the biggest challenges to overcome in SFS. Indeed, as we can see in multiple columns, for a given light direction (white arrow), multiple surfaces can generate the same shading. Image taken from [**61**].

The p,q formulation is commonly used for computing the reflectance, but it is inadequate to represent the surface normal orientation that lies in the xy plane. We will discuss strategies to prevent these problems later.

The primary challenge in SFS lies in resolving the ambiguity of shading (Figure 2.9). This ambiguity arises when different shapes produce an identical shading in an image, making it difficult to ascertain the true 3D structure. Despite ambiguity, most people have expectations about the form that the shape should take. This capability suggests that our visual system employs additional assumptions and cues. Understanding and incorporating the principles behind this human ability to perceive depth from shading is a key area of research in enhancing the accuracy and effectiveness of SFS techniques.

The uniqueness of SFS has been the subject of numerous studies. To our knowledge, no one has been able to prove that SFS is well-posed in the general case. Oliensis [**39, 40, 41**] has shown that a unique solution exists when a surface reflectance is Lambertian and that it is illuminated by a distant light source aligned with the camera. Later, Rouy et al. [**53**] and Lions et al. [**33**] have proven that a solution exists in the viscosity sense for the case where the light is not purely aligned with the camera, when boundary conditions are specified. Prados et al. [**50, 49, 48**] extended the claim for a perspective light (i.e., a point light source) when they relax the conditions on the boundary. To our knowledge, even if the solution has been proven to be unique when boundary conditions are specified, no numerical method has been able to solve SFS robustly.

Algorithms for SFS can generally be classified into four main categories: propagation, minimization, local, and linear. The propagation approach involves propagating information across the domain, starting from initial conditions and gradually extending the solution to the entire area. Minimization methods, on the other hand, are centered around identifying a minimum of energy functions. Local methods operate under the assumption that shapes can be approximated as belonging to a specific subclass within a local region. They simplify the problem by dealing with smaller, more manageable sections of the shape at a time. Linear methods simplify or approximate the equations of SFS by linearizing specific terms. This simplification often makes the problem more tractable, allowing for easier computation. We will now detail each category.

Propagation Methods. One of the first attempts to solve SFS was done by Horn [**20**] through a propagation method. His approach to the problem is to rewrite Equation 2.3.4 and uses the method of characteristics. More on the method of characteristics to solve SFS is provided in Chapter 3.

Alternative propagation methods have been proposed with an emphasis on a specific form of the Hamiltonian-Jacobi equation. Kimmel et al. [**27, 28**] propose a novel approach to solve the SFS problem. By stating that the light is in the same direction as the viewpoint, they reframe the problem as

$$E(x,y) = \frac{1}{\sqrt{1+\frac{dz}{dx}^2+\frac{dz}{dy}^2}} \tag{2.3.5}$$

$$\sqrt{1+\frac{dz}{dx}^2+\frac{dz}{dy}^2} = \frac{1}{E(x,y)} \tag{2.3.6}$$

$$\left|\frac{dz}{dx}^2+\frac{dz}{dy}^2\right| = \sqrt{E(x,y)^{-2}-1}. \tag{2.3.7}$$

The Fast Marching method from Sethian et al. [**56**] and the Level-Set method from Kimmel et al. [**27, 28**] were introduced to solve numerically Equation 2.3.7. Unfortunately, because of the ambiguity of bas-relief shapes, their solution can result in regions where the concavity/convexity is incorrectly assigned. Prados et al. [**49**] generalized the Hamiltonian-Jacobi equation (2.3.7) to regroup orthographic/perspective light configurations into one formulation and provided a numerical solver as well.

Minimization Methods. Instead of using a propagation method, Ikeuchi and Horn [**22**] convert the problem to a functional problem where they retrieve p and q using the following minimization:

$$\min_{p,q} \int_{\Omega} L(x,y,p,q)\, dA = \min_{p,q} \int_{\Omega} (E(x,y) - R(p,q))^2 + \sum_i \lambda_i C_i(x,y,p,q)\, dA \tag{2.3.8}$$

where λ_i is the weight associated to constraint $C_i(x,y,p,q)$. The core component of the minimization is to use variational calculus techniques to find a global solution. However, the nonlinear nature of reflectance makes it difficult to find it. Without any additional constraints, discretizing the problem directly will result in a heavily underconstrained problem and a lack of cohesion between discretized elements, making the surface discontinuous.

To resolve potential discontinuities, Ikeuchi and Horn introduce additional constraints on the solution such as smoothness and occluding boundary conditions

$$C_{\text{smoothness}}(x,y,f,g) = \int \frac{d}{dx} f + \frac{d}{dy} f + \frac{d}{dx} g + \frac{d}{dy} g\, dA \tag{2.3.9}$$

$$C_{\text{boundary}}(x,y,f,g) = \int (f(x,y) - f_{\text{known}})^2 + (g(x,y) - g_{\text{known}})^2 dA \tag{2.3.10}$$

where f_{known} and g_{known} are the values known at the boundary. The f,g coordinates can represent normals that lie in the plane. They are derived from the stereographic projection

$$\begin{aligned} f &\to \frac{2p}{1 + \sqrt{1 + p^2 + q^2}} \\ g &\to \frac{2q}{1 + \sqrt{1 + p^2 + q^2}} \\ p &\to \frac{4f}{4 - (f^2 + g^2)} \\ q &\to \frac{4g}{4 - (f^2 + g^2)}. \end{aligned} \tag{2.3.11}$$

To minimize Equation 2.3.8, Ikeuchi and Horn use the Euler-Lagrange equation, which corresponds to an alternative system where the global minimum is the same as Equation 2.3.8.

An iterative method is used to solve the equation numerically using a finite-difference formulation.

Later, Brook and Horn [**19**] introduce the integrability constraint:

$$C_{\text{integrability}}(x,y,p,q) = \int \frac{dp}{dx} - \frac{dp}{dy}\, dA \tag{2.3.12}$$

to make sure that the surface is integrable and respects the fact that $\frac{dz^2}{dxdy} = \frac{dz^2}{dydx}$. They also propose to include the occluding boundary by using directly the normal $\mathbf{n}$ as a penalty term that circumvents the p,q parameterization. The occluding boundary corresponds to the normal of the shape at the silhouette so those normals are perfectly parallel to the image plane. They use a similar approach to Ikeuchi and Horn to solve their nonlinear system.

Both formulations are interesting, but they offer no guarantee for a global minimum. They seem to produce good results on surfaces with simple curvature.

Instead of directly using the Euler-Lagrange equation, various iterative solvers have been developed for more efficient minimization of the objective functions. Methods such as Gradient Descent and Conjugate Gradient, among others, stand out in this regard. These methods, as discussed in the work of Barron et al. [**1**], Santo et al. [**54**], and Horn [**19**], exhibit distinct convergence properties and have proven to be relatively effective in SFS. However, a notable limitation of these methods is their requirement for extensive manual tuning. Adjusting the coefficients for each constraint can be a meticulous and nontrivial task. Moreover, these adjustments are not consistent across different scenarios, especially when dealing with varying observed intensities in images.

Acker [**11**] introduces the use of a homotopy solver to identify a set of potential solutions for a given observed intensity. His approach yields results qualitatively similar to those obtained by Kimmel et al.'s Fast Marching method [**28**], particularly when the light source aligns with the optical center. When the light is not at the optical center, the method produces wave-like results.

Local Methods. Local approaches to SFS focus on analyzing small patches or points in the image independently to infer local normal orientations. These techniques often use local image gradients and shading information to estimate the normal at each image point. The challenge with local methods is integrating these local estimates into a globally consistent surface, as they can lead to inconsistencies otherwise.

Initially, in 1984, Pentland [**43**] proposed a local approach using local shading analysis by assuming that the shape is locally spherical. However, this assumption holds only for a subset of shapes. Instead, Xiong et al. [**65**] introduce a piecewise method where they solve for each patch locally. By assuming that the solution is locally quadratic and that the light direction is never at the optical center, they can prove that the solution is unique for every patch. Then, they compute the local likelihood distribution associated with each configuration and reuse this information to reconstruct the surface globally. The results of Xiong et al. [**65**] are quite impressive, but they tend to produce more flattened shapes depending on the number of patches that have been used. They also result in concave regions instead of convex regions in specific cases.

Linear Methods. Linear methods simplify the SFS problem by linearizing Equation 2.3.4, which relates the image intensity to the surface orientation and lighting. Pentland [**44**] proposes a similar approach by linearizing the reflectance and using the Fourier transform to retrieve the final shape. Tsai and Shah [**46**] propose a method that approximates p,q by finite difference and linearizes the Lambertian reflectance to retrieve $z(x,y)$. Both methods perform poorly when the surface features large elevations or/and when the light is not aligned with the viewpoint.

A critical observation across the various methods discussed from classical Shape-From-Shading seems that they perform significantly worse if the light is not aligned with the viewpoint. This limitation is consistent, suggesting that the equation for SFS under equal reflectance conditions (Equation 2.3.4) is significantly less effective in such lighting scenarios. In this configuration, the results tend to be greatly influenced by the direction of the light. Moreover, in our work, it is less intuitive for an artist to draw some shading when the light and the viewpoint are aligned in the same direction since it rarely occurs in real-life scenarios.

2.3.2. Shape-From-Shading Related Problems

Since the advent of the original SFS methods, numerous extensions have been proposed.

Shape From X. The field began exploring variants commonly referred to as *Shape From X*, where *X* represents various factors like stereo, motion, texture, focus, and others. In this context, the use of multiple stereo images emerges as a powerful tool for depth retrieval. The redundancy provided by multiple images proves crucial in accurately determining the final shape. In our context, drawing accurate shading in multiple images proves quite a challenge. This is also the case for drawing factors other than shading.

In a similar vein, Barron [**1**] makes a breakthrough by employing statistical inference and incorporating different priors on aspects such as reflectance, shape, smoothness, and color channels. His methodology, utilizing more information than traditional SFS, demonstrates notable results, showcasing the potential of integrating more information from an image into the classical SFS.

introduce NeRF (Neural Radiance Fields) to synthesize novel views. They model a radiance field that represents density and color from different viewpoints. To retrieve the radiance field, they train a neural network on a coherent collection of images from various viewpoints. Their method can also be used to extract shape information.

Despite their results, NeRF-based methods come with notable limitations. The core issue lies in the volumetric representation they rely on, which, while powerful, can sometimes restrict broader applicability, particularly in dynamic or complex environments. Furthermore, these methods require significant computational resources. It is particularly relevant in fields like real-time gaming, virtual reality, or augmented reality, where speed is as crucial as accuracy. It is also problematic due to the sheer volume of input data needed. Requirements of large data are crucial to effectively resolve ambiguities in the scene and reproduce it accurately. A recent development by Kerpb et al. [**25**] using a rendering technique called Gaussian Splatting shows significant improvements in training speed, but it is still not feasible for real time.

Differential Rendering. In the computer graphics community, based on recent breakthroughs in machine learning [**30**], a renewed emphasis has made differentiable tools more accessible. This shift has notably influenced optimization methods, with minimization emerging as a de-facto tool for various tasks. Of particular interest are recent developments in differential rendering [**68, 38**].

The task is to minimize the rendered output over an objective function

$$\min_{\mathbf{x}} \sum_i J(I(\mathbf{x}_i))$$

where I is the rendered image, $\mathbf{x}$ are the scene parameters encoded as a vector, and J is the objective function. The difficulty of using such a tool comes with computing the Jacobian matrix of $I(\mathbf{x})$, where computations can be intractable. The main focus of differential rendering research is to make such computations feasible with limited resources.

Even though this method has shown potential, in its current state, it suffers from many drawbacks such as heavy computations and a reliance on multiple images as targets to converge to a proper solution. In our work, the costly computations made it impractical for a real-time application and have been avoided. However, it was deemed important to acknowledge based on recent developments.

2.4. 3D Creation

In computer graphics, 3D creation is the process that regroups the elaboration of virtual elements in a scene. It is used extensively across multiple disciplines. In manufacturing, 3D creation is crucial for realistically mimicking objects. It enhances accuracy, reduces costs, improves safety, and allows for various optimizations and simulations to identify potential design flaws early. In the medical field, it is used for detailed replication of organs, aiding in measurement, study, and treatment planning. Architects leverage 3D creation to visualize and refine their spatial designs before physical construction, offering a preview of their final product with simulated global illumination. In the entertainment industry, the focus shifts to creating either artistically appealing, hyper-realistic, or imaginative scenes. These settings can blend virtual elements seamlessly into live-action sequences, and bring to life scenes that might be impractical or impossible to physically construct, including historical or fictional environments.

In the entertainment industry, the process can be roughly divided into three key areas: 3D modeling, animation, and composition. 3D modeling is about crafting shapes that are either accurate representations or visually striking, using advanced 3D software. Animation creates the illusion of movement through the display of sequential images. It often incorporates tools to simplify complex phenomena like fluid dynamics, soft-body physics, cloth and hair simulations, and facial expressions. Composition in computer graphics, much like in cinema, involves arranging elements within a scene from a specific viewpoint. This includes manipulating camera parameters and lighting effects to achieve a desired style or mood. The final appearance of a scene is a harmonious blend of these elements.

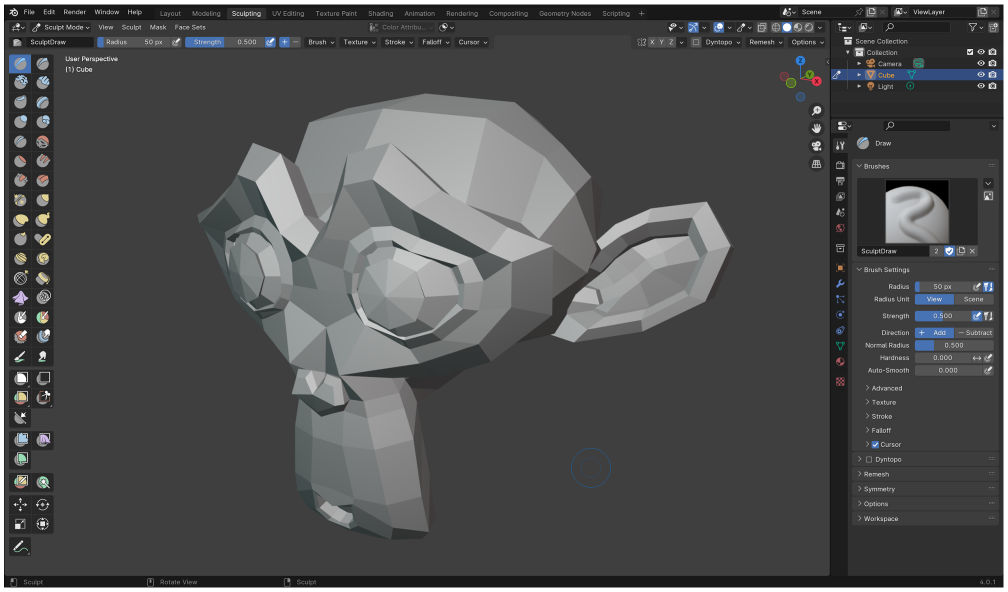


Figure 2.10 – Example of a typical professional 3D modelling tool (Blender). 3D modeling is a powerful tool for editing shapes, but the steep learning curve makes it harder for beginners and even intermediate users because of the numerous manual interactions.

Our focus will be on the creation of static 3D shapes, particularly based on triangle mesh structures. 3D modeling, in essence, involves designing an object to meet a specific outcome by modifying its shape and attributes.

3D modeling is a process where users work with all kinds of inputs and algorithms to create interesting shapes. The design of a mesh is a task requiring skilled artists. In professional 3D software like Maya, 3DS Max, and Blender, the initial step usually involves creating or acquiring the basic structural components of a mesh. This step includes generating vertices and triangles, subdividing triangles, and displacing vertices.

Often, a simpler primitive shape serves as a basis for further development. Artists refine the structure, reshaping sections or regions in a manner akin to clay modeling. This involves techniques to enlarge, reduce, or alter surface areas. Achieving high-quality results requires a blend of intuition and iterative design to use most efficiently the tools available.

Subsequent steps may involve rigging, a process to prepare the shape for animation within 3D modeling software. Rigging creates an underlying skeleton that is simpler to animate and enables believable deformations for the original geometry. Rigging thus simplifies the animation process. Artists may also need to adjust the mapping of 2D or 3D textures over a surface to add finer details and minimize visual distortion due to the pixelization of textures.

The inclusion of physics-based modifications, like cloth deformations, hair dynamics, and soft-body physics, adds a layer of realism but also complexity. These aspects are challenging to fine-tune and demand significant time and expertise.

Throughout these stages, artists must continually change their viewpoint, examining the geometric model from different angles to ensure accuracy and aesthetic appeal. Each task in a 3D modeling process is intricate and demands a high level of skills and attention to detail.

Another paradigm to reduce this burden is procedural content generation. Procedural content is a procedure or an algorithm where a set of pre-established rules are used to generate complex models. The Lindenmayer system [**32**] is a procedural method, often called L-System, that uses symbols as input and substitution rules to generate locally new shapes. It is quite popular for creating various types of trees as demonstrated by Boudon et al. [**6**]. Perlin noise [**45**] is a type of procedurally generated noise popular for self-similar properties. A common usage is to imitate atmospheric or geologic phenomena such as cloud formations in the sky, caves, island formations, etc. In the last few years, Generative AI has become much more prevalent because of its ability to build an implicit set of rules learned from data to generate content from images or texts. In all these examples, procedural modeling frees the artists from tedious modeling, but it offers very little control to modify the results, apart from a complete regeneration.

In our work, we will concentrate on a unique approach to 3D modeling that draws inspiration from sketching, and more specifically on geometric approaches. Sketching represents a more intuitive natural process, making it an appealing paradigm for simplifying the workflow for users with less experience. This approach leverages the simplicity and immediacy of sketching, translating these qualities into the 3D modeling process.

Igarashi et al. [**21**] introduced a framework called *Teddy* to create a 3D shape using stroke-based input. The user draws a contour where an internal shape is inflated. The system enables cutting, bending, and merging shapes to create casual complex objects with great simplicity. Li et al. [**31**] introduced *Bendsketch*, a framework where the user uses annotated strokes to build a directional vector field inside a contour. Instead of directly altering the shape, their system constructs a principal direction field by aligning the directional vector field with the bending strokes. In an iterative scheme, they spread the principal curvature over the directional field. Then, they minimize the energy to lift the shape in 3D. They also introduce different types of stroke discontinuities, sharp features, ridges, valleys, and flat

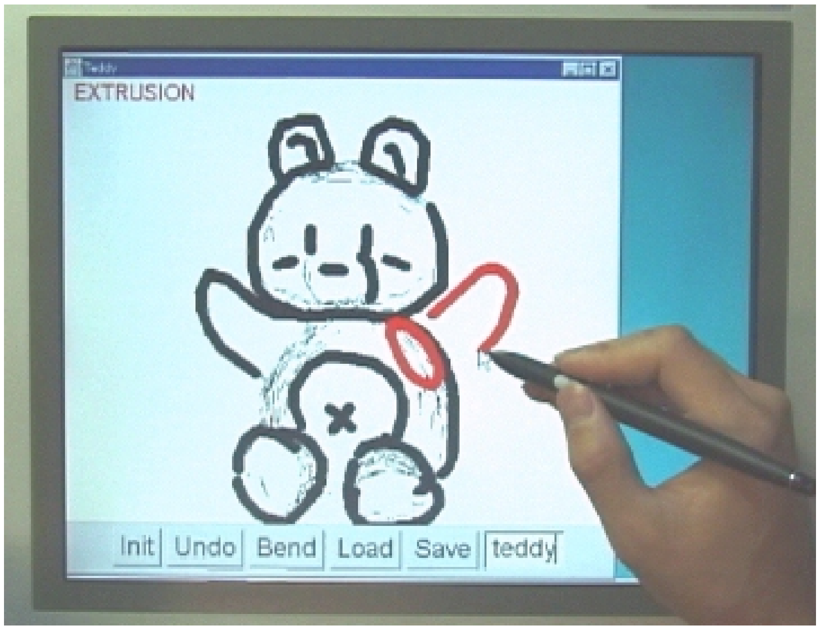


Figure 2.11 – Sketch-based modeling where the user draws a region on a screen. A 3D geometric model is generated from it. Image taken from [**21**].

strokes to improve their results. Dvoroznak et al. [**9**] use a framework where they rely on regions that are glued together as one mesh. The mesh is inflated. Each region is shifted so that the mesh does not self-intersect while ensuring various boundary conditions to make it seamless. In a follow-up, Dvoroznak et al. [**10**] propose *Monster Mash* using an adapted ARAP algorithm [**57**] instead of shifting each region apart. Our work will extend *Monster Mash*; we go more in-depth in Chapter 3.

More precisely in sketch-based modeling, shading-based methods for editing 3D models have received limited attention from the research community despite shading being a natural concept to artists. Shading-based methods leverage the shading information from a shape to edit or reconstruct a 3D model. Gingold et al. [**16**] introduce shading-based surface editing using a specific set of strokes to modify regions and accentuate shading effects by bending regions and moving highlights. However, their tool only allows a user to slightly alter a small region of the shape. Xu et al. [**66, 67**] introduce an interactive method where a user draws an isophote that reaches at least one contour of a shape. Based on the surface normal at the contour, they interpolate the normals along the isophote while remaining on the cone formed by the Lambertian reflectance. However, their method presents some limitations. It requires the user to draw multiple isophotes and to manually edit the interpolated normals in order to produce visually accurate results, which is quite hard to do, even for an artist.

In our work, we will focus on a method based on the method of characteristic strips from SFS to retrieve a final shape by manipulating directly the shading.

Besides creating or editing the global look of a shape, some research has been applied to decorate a surface with high-frequency details. Fanni et al. [**13**] use an algorithm to embellish a shape by generating a 3D volumetric pattern around a surface. A procedure stacks the volumetric patterns together and produces a visually appealing result similar to pottery. Instead of relying on texture mapping to produce higher frequency details on a mesh, Nazzaro et al. [**37**] generate a pattern directly in the intrinsic space of the shape. To enable interactive modification on the surface shape, they propose a new algorithm to find the geodesic path on a surface so a user can edit directly on the shape. Their approach produces interesting patterns by avoiding distortion commonly happening from 2D texture mapping, but requires a large number of vertices and triangles. Mancinelli et al. [**34**] introduce an interactive Bézier spline tool on the surface of meshes with a large number of vertices and triangles. It is based on an alternative approach to Casteljau and Bernstein evaluations. Riso et al. [**52**] create a procedure where a user draws 2D polylines directly on the intrinsic surface of a mesh and uses boolean operations in the intrinsic surface.

In this literature review, we covered multiple facets of sketch-based modeling without mentioning learning-based approaches. Learning-based approaches have become more popular in the last decades due to their powerful versatility. However, learning-based approaches have often the same limitations. Indeed, they are great for end-to-end pipelines, but altering partial regions can lead to non-intuitive effects on the rest of the shapes. They are often poorer to control than traditional tools. For those reasons, we limited our literature review to non-learning-based approaches.

Chapter 3

Background

In this chapter, we provide additional information about the approaches we used for our method. We elaborate on the method of characteristics in the context of SFS (Section 3.1), and the method from *Monster Mash* by Dvoroznak et al. [**10**] for casual 3D modeling (Section 3.2).

3.1. Method of characteristics for SFS

Horn [**20**] uses the method of characteristics to solve SFS. Equation 2.3.4 is rewritten as

$$F(x,y,p,q,z) = E(x,y) - R(p,q) = 0 \tag{3.1.1}$$

where $z|_\Gamma = \phi|_\Gamma$ for an initial curve Γ. Equation 3.1.1 is considered fully nonlinear. Without further knowledge about the shape, we can only require E to be differentiable.

The method of characteristics states that we can reformulate the equation as an ODE system of strips $(x(s), y(s), p(s), q(s), z(s))$

$$\begin{aligned}
\frac{dx}{ds} &= \frac{d}{dp}F = -\frac{d}{dp}R \\
\frac{dy}{ds} &= \frac{d}{dq}F = -\frac{d}{dq}R \\
\frac{dp}{ds} &= -\frac{d}{dx}F - p\frac{d}{dz}F = -\frac{d}{dx}E \\
\frac{dq}{ds} &= -\frac{d}{dy}F - q\frac{d}{dz}F = -\frac{d}{dy}E \\
\frac{dz}{ds} &= p\frac{d}{dp}F + q\frac{d}{dq}F = p\frac{d}{dp}R + q\frac{d}{dq}R
\end{aligned} \tag{3.1.2}$$

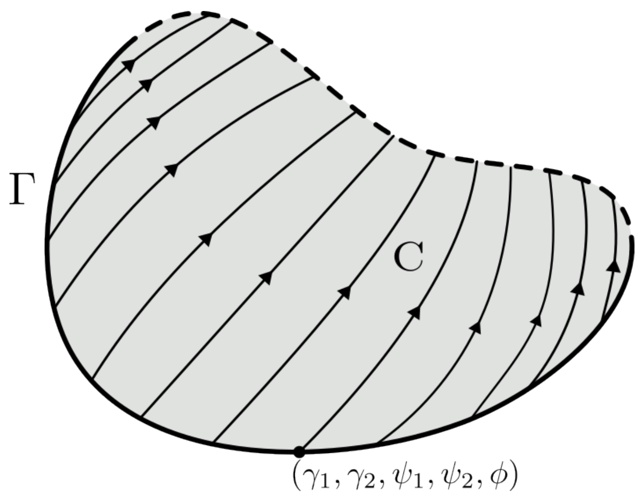


Figure 3.1 – Flow of strips C (thin black curve) from the method of characteristics starting from an initial curve Γ (thick black curve). The black dot corresponds to an initial point $(\gamma_1, \gamma_2, \psi_1, \psi_2, \phi)$ from Γ. The dashed curve delimits the domain.

where we need to specify five initial conditions at the same initial curve Γ for characteristic strips:

$$\begin{aligned} x|_\Gamma &= \gamma_1|_\Gamma \\ y|_\Gamma &= \gamma_2|_\Gamma \\ p|_\Gamma &= \psi_1|_\Gamma \\ q|_\Gamma &= \psi_2|_\Gamma \\ z|_\Gamma &= \phi|_\Gamma. \end{aligned} \tag{3.1.3}$$

Figure 3.1 illustrates characteristic strips for an initial curve Γ.

We can interpret the orientation of a strip as two vectors in the intensity and reflectance spaces. The gradient of the reflectance space is used in the intensity space and vice versa. In itself, the gradient of reflectance R represents the direction of changes in brightness in the tangent plane. Meanwhile, the gradient of intensity E is related to the direction of changes in brightness from a given viewpoint.

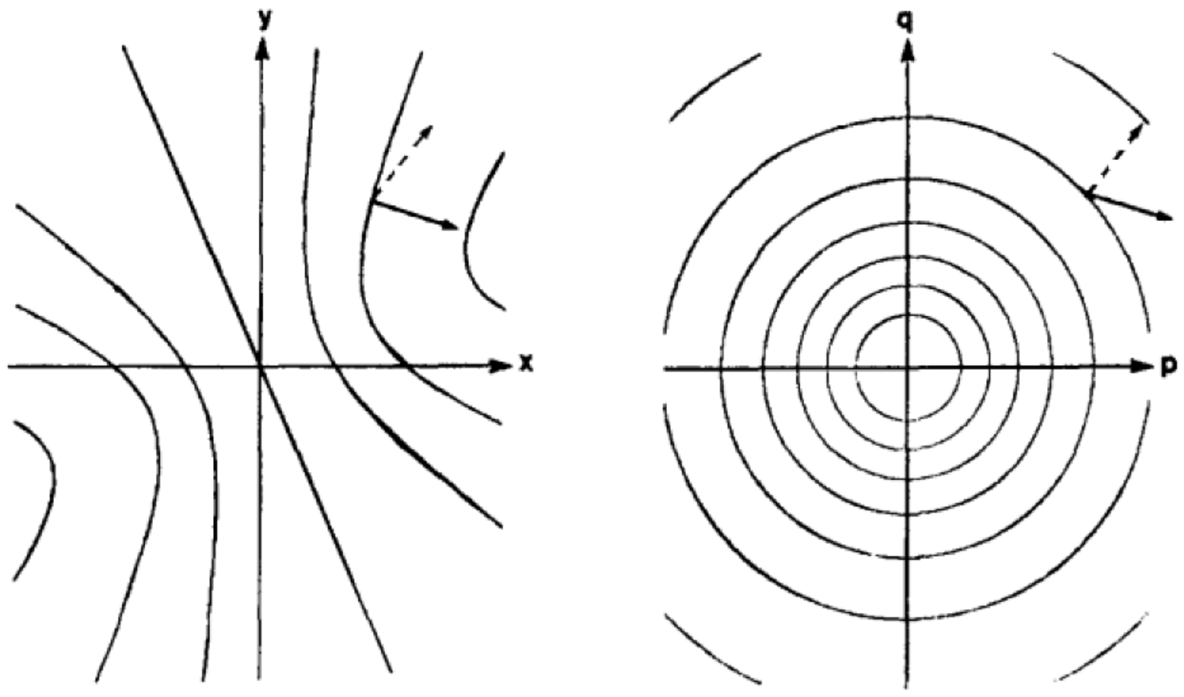


Figure 3.2 – Geometric interpretation of the method of characteristics for SFS. On the left, the plot depicts the intensity E. On the right, it depicts the reflectance R. Dashed arrows represent the gradient for the reflectance while full arrows represent the gradient in the intensity plane. Image adapted from [**19**].

The conditions at the boundary need to be admissible, which requires one point on the initial curve $x_0,y_0 \in \Gamma$ to respect

$$\begin{cases} F(\gamma_1,\gamma_2,\psi_1,\psi_2,\phi)|_\Gamma = 0 \\ \psi_1|_\Gamma = \frac{d}{dx}\phi|_\Gamma \\ \psi_2|_\Gamma = \frac{d}{dy}\phi|_\Gamma. \end{cases} \tag{3.1.4}$$

If all the points at the boundary are admissible and there exists an initial curve that is non-characteristic such that

$$\begin{bmatrix} -\frac{d}{dp}R & -\frac{d}{dq}R \end{bmatrix} \cdot \begin{bmatrix} -\frac{d}{ds}\gamma_2 & \frac{d}{ds}\gamma_1 \end{bmatrix} \neq 0 \tag{3.1.5}$$

a local solution exists for the strips near Γ.

We can go further than Horn [**20**] in analyzing the stability of Equation 3.1.2. Equation 3.1.2 can be characterized as a nonlinear autonomous system $\frac{d\mathbf{x}}{ds} = f(\mathbf{x}(s))$ where $\mathbf{x} = (x,y,p,q)$ in our specific case. The system has no explicit dependency on s. Let us assume that only one peak of brightness ($R(p_e,q_e) = 1$ and $E(x_e,y_e) = 1$) is possible. At the peak intensity in the image $\mathbf{x}_e$, we can trivially show that $f(\mathbf{x}_e) = 0$ because the gradient of E and R will be 0 at the equilibrium point $\mathbf{x}_e$.

To find if the autonomous system is Lyapunov stable, we need to find a Lyapunov function that will respect the following conditions

$$\begin{cases} V(\mathbf{x}) = 0 & \text{where } \mathbf{x} = \mathbf{x}_e \\ V(\mathbf{x}) > 0 & \text{where } \mathbf{x} \neq \mathbf{x}_e \\ \nabla \mathbf{V}(\mathbf{x}) \cdot \frac{d\mathbf{x}}{ds} \leq 0 & \forall \mathbf{x}. \end{cases} \tag{3.1.6}$$

Let us consider the following function

$$Q(x,y,p,q) = 2 - R(p,q) - E(x,y). \tag{3.1.7}$$

We seek to determine whether it is Lyapunov stable when there is a peak present in the observed intensity.

At the equilibrium point, we can verify the first condition by evaluating $Q(x,y,p,q) = 2 - R(p_e,q_e) - E(x_e,y_e) = 2 - (1) - (1) = 0$. Because there is only one peak, Q is bounded by $0 \leq Q \leq 4$ and respects the second condition.

For the third condition, the intensity E is problem-dependent. We cannot predict how it will behave. Instead, we will seek to find a way to respect the condition without requiring an explicit function for it by assuming that E is defined and smooth.

$$\nabla Q(\mathbf{x}) \cdot \frac{d\mathbf{x}}{ds} \leq 0 \tag{3.1.8}$$

$$-\frac{dR}{dp} \cdot \left(-\frac{dE}{dx}\right) - \frac{dR}{dq}\left(-\frac{dE}{dy}\right) - \frac{dE}{dx} \cdot \left(-\frac{dR}{dp}\right) - \frac{dE}{dy} \cdot \left(-\frac{dR}{dq}\right) \leq 0 \tag{3.1.9}$$

$$2\frac{dR}{dp} \cdot \frac{dE}{dx} + 2\frac{dR}{dq} \cdot \frac{dE}{dy} \leq 0 \tag{3.1.10}$$

$$\frac{dR}{dp} \cdot \frac{dE}{dx} + \frac{dR}{dq} \cdot \frac{dE}{dy} \leq 0 \tag{3.1.11}$$

The solution will be at least Lyapunov stable in the region around the equilibrium where the dot product between the gradient of R and the gradient of E at $\mathbf{x}$ is negative. In the presence of multiple peaks, we can make the standard claim that the solution will be Lyapunov stable locally around each peak.

The method of characteristics [1] is particularly compelling due to its inherent ease of implementation. The characteristics strips are therefore well-suited for real-time applications. However, a primary limitation of these techniques lies in their reliance on the accuracy of initial values. This is because errors present at the onset tend to propagate throughout the integration process, impacting the overall quality of the outcome.

3.2. *Monster Mash* Approach

In *Monster Mash*, Dvoroznak et al. [**10**] propose an algorithm to transform curves drawn over separate layers by a user into a pleasant-looking mesh. In their original work, each curve lie in a layer and the image plane. The user can alter the depth of each layer through a keyboard input. Each curve Γ_{contour} delimits a domain Ω where the line that connects two endpoints corresponds to a special curve call Γ_{conn}. The interiors of each Ω are triangulated through Delaunay triangulation by adding additional vertices in the interiors. We refer to those triangulated interiors as faces. Each face is a triangle mesh with an orientation relative to the image plane. Each layer has one front face and one back face. The front face and back face are nearly identical. However, the vertices of each back face are ordered in reverse (i.e., clockwise) to have their normals pointing in the opposite direction, and their height flipped.

Stitching. Each pair of layers where Γ_{conn} lay over another Ω is stitched together. Then, they insert the vertices from the face under Γ_{conn} into the other face. Depending on if the layer is below or above, they stitch the vertices respectively with the front face or the back face. A hole is made in the other face to keep it manifold. Triangles are then created from the vertices under Γ_{conn} and the one where the vertices has been inserted. To stitch the front faces and back faces into one mesh, new triangles are formed with the vertices under Γ_{contour} of each front face and back face (Figure 3.4).

Inflation. The mesh is inflated based on the method of Sykora et al. [**60**]. Sykora et al. [**60**] use a linear system to retrieve the height:

$$\begin{aligned} \mathbf{L}\mathbf{z}^2 &= \mathbf{k} \\ \text{s.t. } \mathbf{z}_{\text{bnd}} &= 0 \end{aligned} \tag{3.2.1}$$

where $\mathbf{L}$ is the cotangent Laplacian, $\mathbf{k}$ is the coefficient of inflation, and $\mathbf{z}$ is the height for the vertices of each face. Two solutions are possible. In *Monster Mash*, they keep the one

1. For more detailed information on the method of characteristics, please refer to the book of Evans [**12**].

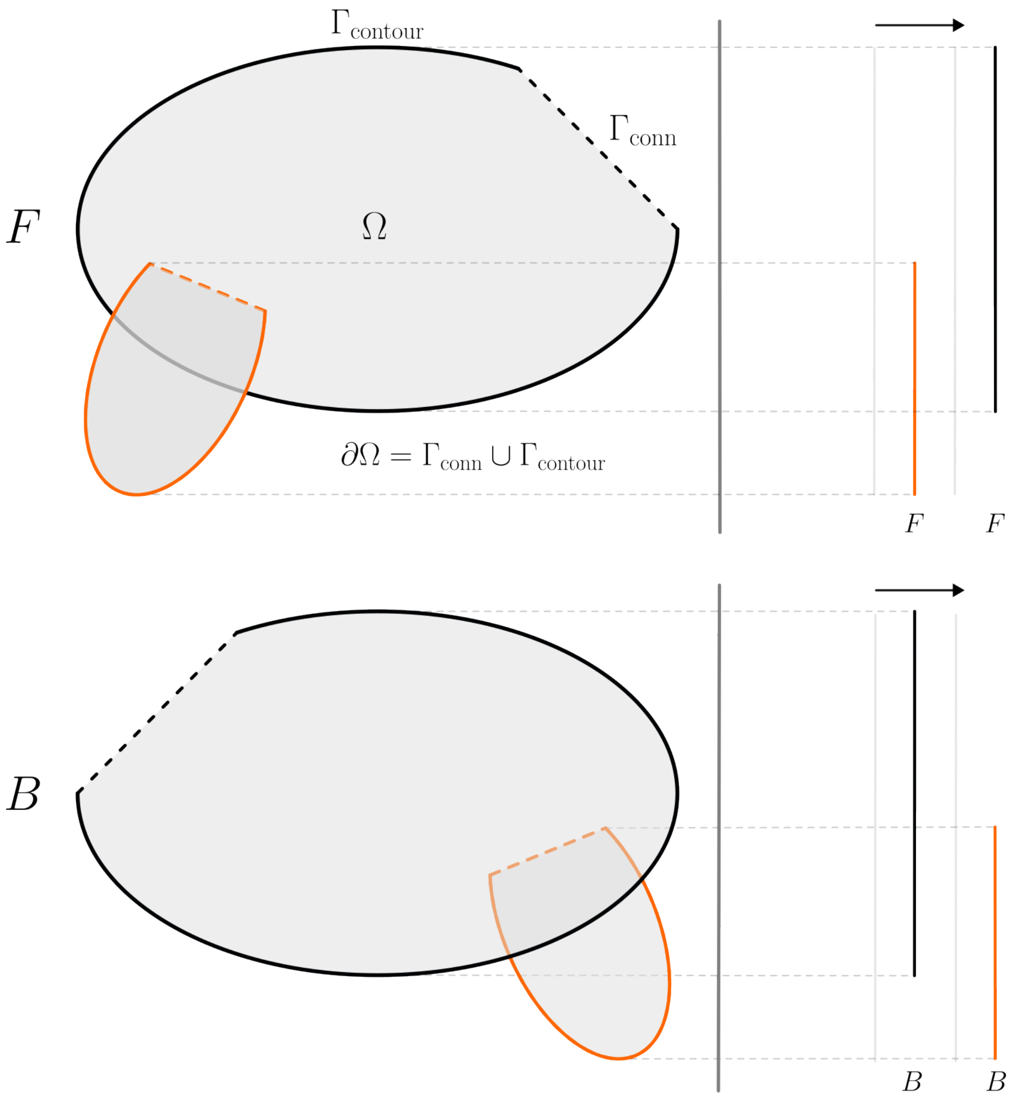

Γ_{contour}
Γ_{conn}
F
Ω
$\partial\Omega = \Gamma_{\text{conn}} \cup \Gamma_{\text{contour}}$
F
F
B
B
B

Figure 3.3 – Two faces are displayed, one with an orange contour and one with a black contour. Each color is associated with a specific layer. A layer has two faces: front and back. We display each face as a flat mesh for illustrative purposes. On the left, we can see the front F (top) and back B (bottom) faces. We denote in black the contour Γ_{contour} of region Ω. A dashed line Γ_{conn} represents the segments of the shape that will be connected to other layers. On the right, we emphasize their relative order from an alternative viewpoint. A dashed segment Γ_{contour} represents a segment that could potentially be merged between two faces.

where the direction of $\mathbf{z}$ inflates toward the same direction as the face. Then, they take the square root of the solution. It results in a semi-elliptical-like shape with a pleasing variation in height except close to the boundary. We propose an improvement to make the transition smoother from the boundary in Section 4.2.

Deformation. The mesh is deformed with a modified As-Rigid-As-Possible (ARAP) algorithm to shift each vertices associated with a layer so that vertices at different layers do not overlap. Originally, ARAP was introduced by Sorkine and Alexa [**57**]. The model can be stated as

$$\frac{1}{2}\min_{\mathbf{v},\mathbf{R}}\sum_{i}\sum_{j\in\mathcal{N}(i)}||(\mathbf{v}_i-\mathbf{v}_j)-\mathbf{R}_i(\mathbf{v}_i^0-\mathbf{v}_j^0)||^2 \tag{3.2.2}$$

where the energy is minimized through a local pass and a global pass. In the local pass, they minimize the objective by letting the rotation matrix $\mathbf{R}$ free and freezing vertices $\mathbf{v}$. In the global pass, they repeat the process, but they let $\mathbf{v}$ free and freeze $\mathbf{R}$.

Dvoroznak et al. [**10**] introduce new inequality constraints into ARAP to shift vertices by their respective layers:

$$\begin{aligned}\mathbf{z}_i &\geq \mathbf{z}_j \quad \forall(i,j)\in C^{\geq}\\ \mathbf{z}_i &\leq \mathbf{z}_j \quad \forall(i,j)\in C^{\leq}\\ \mathbf{z}_i &= \mathbf{z}_j \quad \forall(i,j)\in C^{=}\end{aligned} \tag{3.2.3}$$

$C^{\geq}$ is the set of vertices underneath Γ_{contour} for the back faces that are above the nearest vertices of the back faces below. $C^{\geq}$ is the set of vertices underneath Γ_{contour} for the front faces that are above the nearest vertices of the front faces below. $C^{=}$ is the set of vertices under Γ_{conn} from the front face that should be at the same position as the vertices from the below back face under Γ_{conn} (Figure 3.5). Other constraints can also be added to animate the final mesh, but are off-topic to our method.

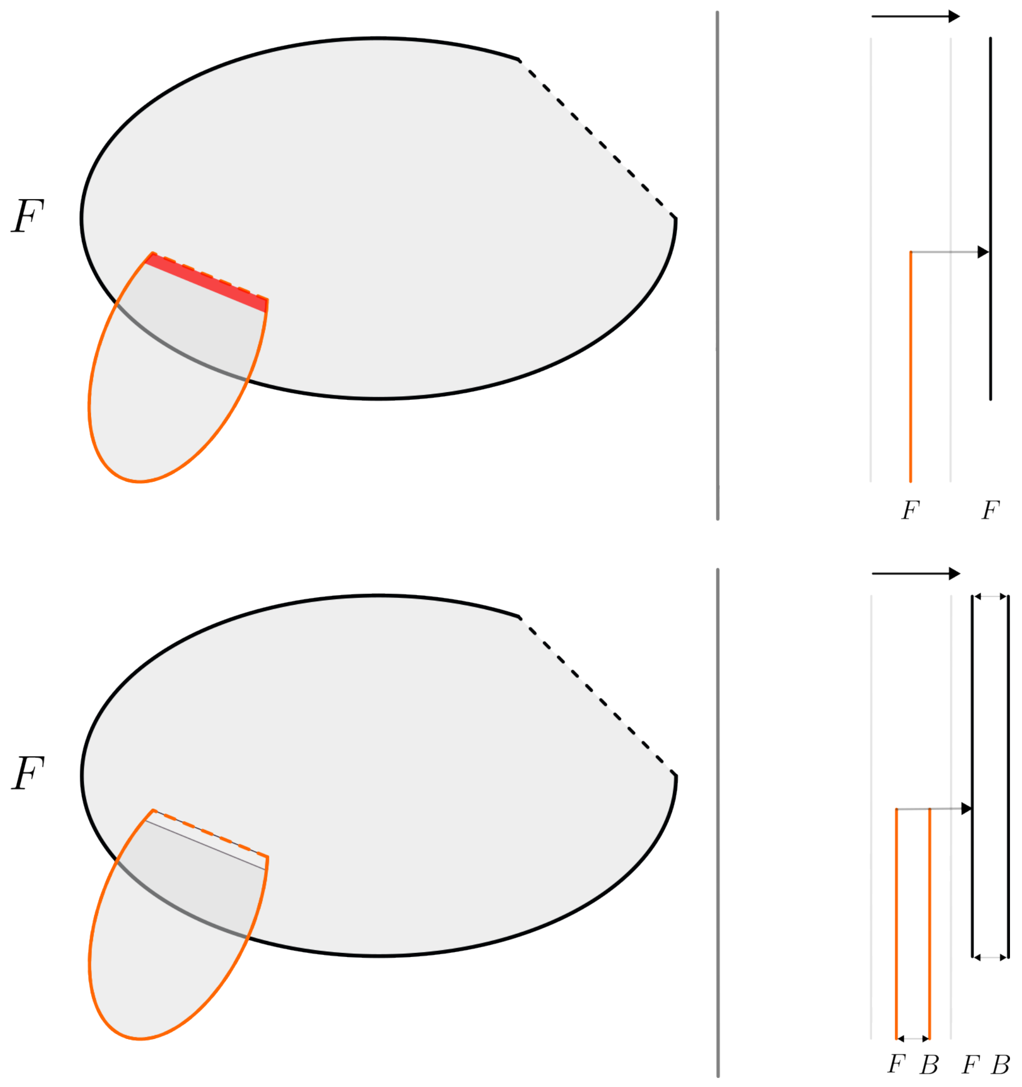
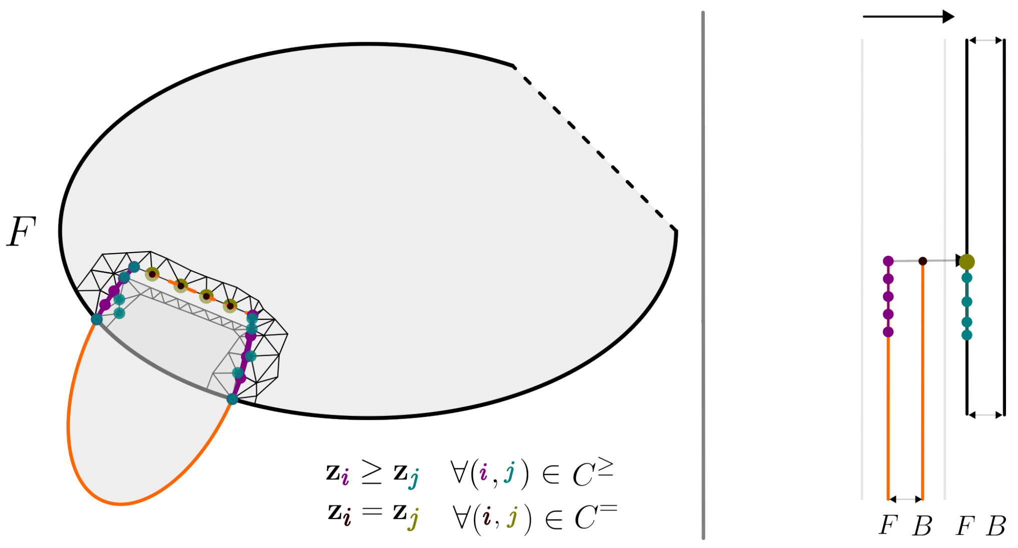


Figure 3.4 – On the top, we illustrate the stage where we stitch faces by their Γ_{contour}. On the bottom, we illustrate how we stitch the front and back faces together. On the top left, the orange upper face is connected with the black face below at the dashed line Γ_{contour}. The vertices in the hole in red (thicker segment) have been removed to keep the mesh manifold. On the top right, the orange front face is connected to the black front face. The black arrow illustrates the connection between the two faces. No stitching is needed for back faces. On the bottom left, the orange and black front faces are displayed in the same order. On the bottom right, the front and back faces are connected through the black double-headed arrows. The orange back face and black front face are not connected under Γ_{conn}. Otherwise, the mesh would be non-manifold.



F
$\mathbf{z}_i \geq \mathbf{z}_j \quad \forall (i, j) \in C^{\geq}$
$\mathbf{z}_i = \mathbf{z}_j \quad \forall (i, j) \in C^{=}$
F B F B

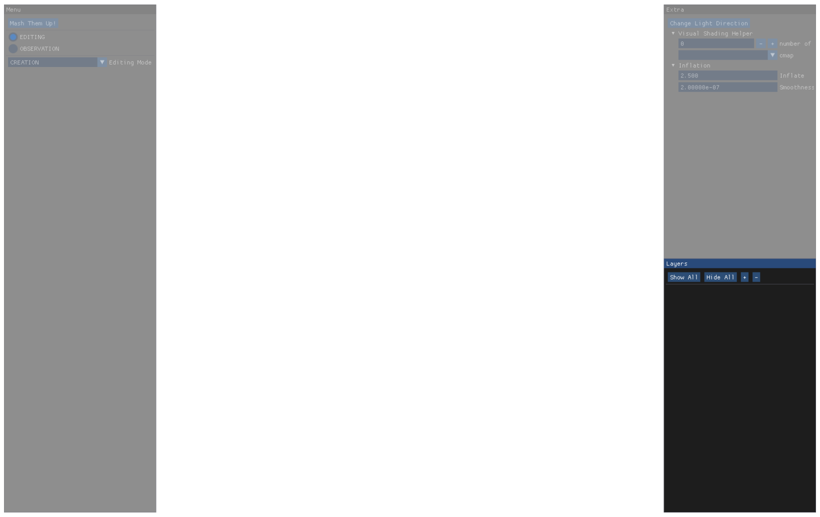
Figure 3.5 – On the left, the dots illustrate the constraint on the vertices that are part of $C^{\geq}$ and $C^{=}$. $C^{\geq}$ corresponds to the set of constraints where the z components of the orange back face are above the z components of the black front face. $C^{=}$ is the set of vertices under Γ_{conn} from the front face that should be at the same position as the vertices from the below back face under Γ_{conn}. For a vertex on the contour of the orange face, we pick the nearest vertex in the face below. On the right, the same dots illustrate how we impose the constraints on the orange back face and the black front face.

Chapter 4

Methodology

In this chapter, we start with a short overview of our system (Section 4.1). Then, we start elaborating on our method. After the planar mesh has been created by the user with our system, the mesh is inflated using the method from Dvoroznak et al. [**9, 10**] (Section 4.2). To isolate each change without affecting the rest of the mesh, we start by identifying regions of the original mesh using the strips from the method of characteristics (Section 4.3). We call it the impacted region. The region can extend outside of the band as it depends on the characteristic strips. Then, we seek to use the method of characteristics to reconstruct the final height. To proceed, we must define the resulting shading from redrawing the isophote. We interpolate the new shading by fixing the value at the redrawn isophote and at the boundary of the region common to the impacted region and the band (Section 4.4). The method as described by Horn [**20**] is used to retrieve the height from our newly interpolated shading (Section 4.5). The penultimate step involves the reconstruction of height from the integrated strips (Section 4.6). Inspired by *Monster Mash* from Dvoroznak et al. [**10**], at the end, we assemble each face into a single one (Section 4.7) using the relative order of the layers being displayed.

Because we are working in a virtual environment, we can substitute the BRDF with an equivalent function that accepts negative values. The consequence is that the gradient of the reflectance is thus always defined at any point. In addition, we only support the configuration where the light is orthogonal to the viewpoint direction. In other words, the light can only lie in the plane orthogonal to the camera.



CREATION Editing Mode

0 number of

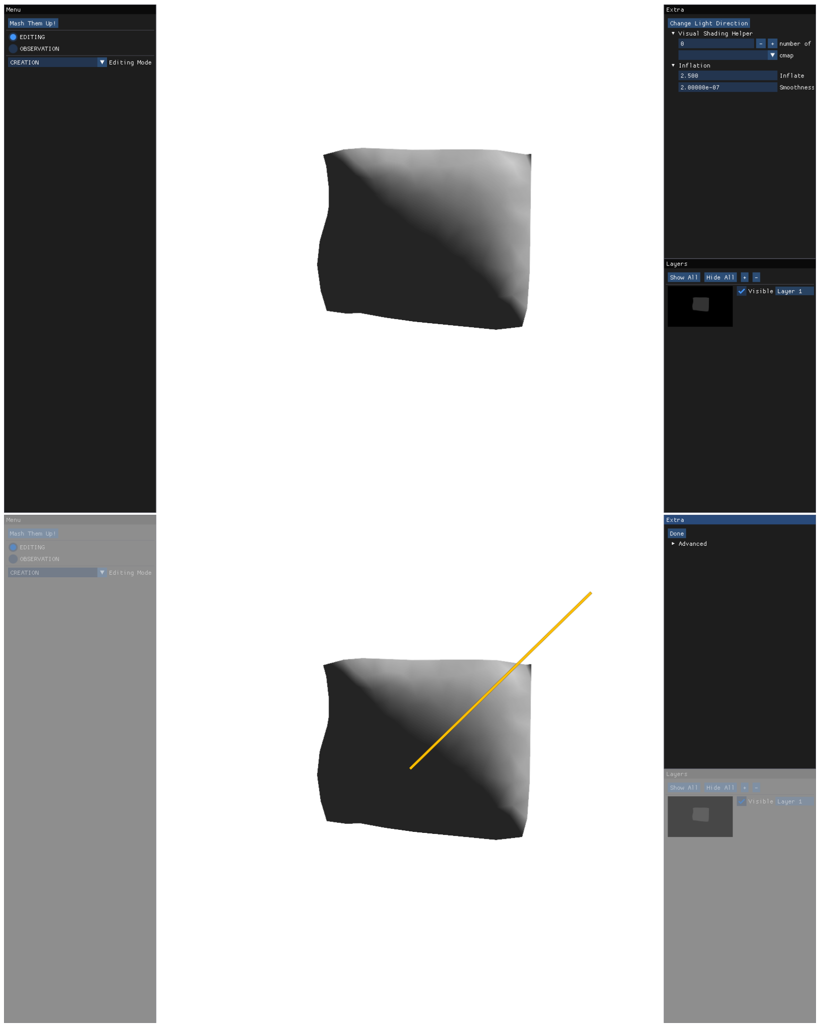
2.500 Inflate

2.00000e-07 Smoothness

Figure 4.1 – State of the application at launch. The user add one canvas on which he or she can draw a contour. The panel on the left corresponds to the menu. The panel on the top right corresponds to the various controls for the mesh and the lighting direction. The panel on the bottom right displays the ordered layers of canvases.

4.1. System Overview

The application launches with three panels and a drawing region in the center, as shown in Figure 4.1. The bottom right panel lets users manage layers. Layers can be added and removed using the "+" and "-" buttons respectively. Each layer has a canvas on which a user can draw in the central drawing region. A miniaturized canvas is displayed for each layer. Each canvas can have only one polyline that describes the contour of a mesh. The user can choose to hide a selected layer, which hides anything related to that layer in the drawing region. This makes it less distracting for the user when drawing on another canvas. The panel on the left is the main menu. The radio buttons refer to the two modes for our application. The "EDITING" mode activates the drawing region where the user creates and edits meshes. In "EDITING" mode, the panel on the top right offers extra options to the user. The "OBSERVATION" mode lets the user visualize the mesh in a 3D environment through rasterization in the center of the screen where the drawing region is. To improve visibility, we display the mesh with a flat shading that does not interpolate normals between vertices; it uses the face normal instead. The user can move the camera around by using the mouse to orbit around his creation. When he selects the "SCULPTURE" state, the user can choose in the panel on the top right between various tools to edit the mesh. Currently, we only support one tool which corresponds to our shading-based method.



CREATION Editing Mode
0 number of
cmap
Inflation
2.500 Inflate
2.00000e-07 Smoothness

Figure 4.2 – (Top) State of the application after having drawn a contour. A shaded mesh is displayed in the drawing region. (Bottom) State of the application when changing the light direction. The light direction can be changed by clicking down and moving the mouse. The shading is properly updated by taking the segment between the mouse position and the center of the drawing region. The light direction can only lie in the plane of the drawing region.

At first, in "CREATION" state, the user adds one layer and draws a contour on the associated canvas. The interior of the contour is triangulated into a mesh using the *triangle* library. The first two coordinates of a vertex of the mesh represent the position of the vertex in the image plane while the third coordinate represents its height from the image plane. The image plane corresponds to an arbitrary plane in front of the camera where the normal of the plane is aligned with the viewpoint. Each layer shares the same image plane. The newly created mesh is associated with the layer and is inflated. The user can tweak the inflation using the

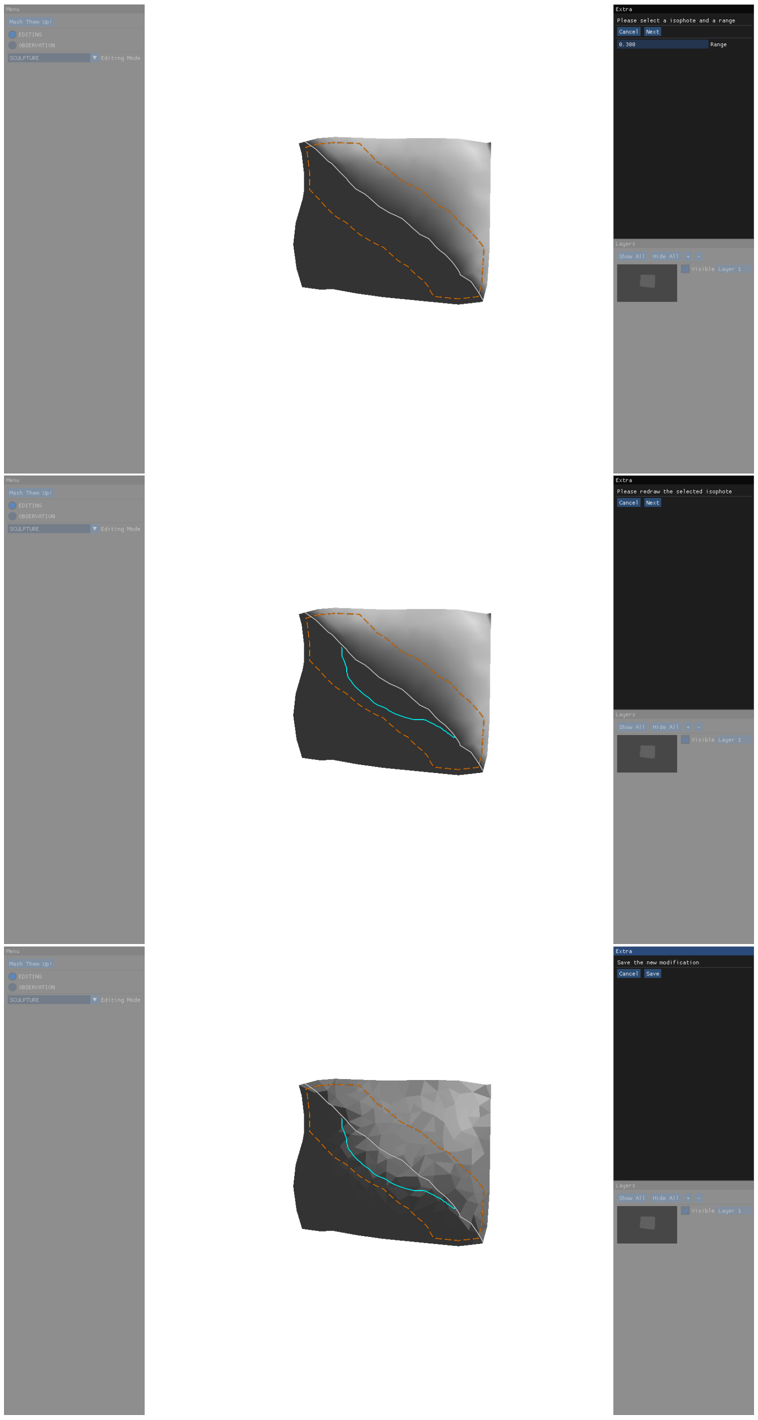


Figure 4.3 – (Top) State of the application after selecting the "SCULPTURE" state at each step of the process using our system. (Top) The first step corresponds to choosing an isovalue by clicking on the mesh and defining a range for the band in dashed orange. A white isophote highlights the selected isovalue. (Middle) The user redraws an isophote within the band. By experimenting with our system, the user gains knowledge about how redrawing the isophote can affect the overall height. (Bottom) The edited mesh is displayed while the user can move the camera around it.

top right panel in "CREATION" state. The newly created mesh is displayed at the same position as the contour in the drawing region. The user can change the light direction by clicking down and moving his mouse around the center of the drawing region after selecting the corresponding button (Figure 4.2).

The shading editing tool is accessible through the "SCULPTURE" state (Figure 4.3). The process begins by selecting a pixel in the image plane. The corresponding point in 3D has an associated isophote value (i.e., $\mathbf{n} \cdot \mathbf{l}$ as explained in Section 2.3.1). The associated isophote curve is displayed in white within a band surrounded by an orange dashed curve. The isophote is extracted as a polyline on the triangle mesh, assuming that the three vertices of a triangle have different normals. To retrieve it, we assume that the isovalue at each vertex is used to interpolate isovalues across triangles using barycentric interpolation. If the isophote traverses a triangle, it will intersect two of its edges. We compute the two points by linear interpolation. The final isophote is encoded as an ordered list of the intersection points along the edges of triangles. It is stored as a polyline that we call the selected isophote. In our method, the shading can only be manipulated by directly editing isophotes. Other drawing techniques, such as pointillism, are avoided because they require more advanced artistic knowledge about shading. By minimizing the number of inputs that a user needs by editing isophote directly, we aim to make our tool easier to use and more accessible for casual 3D modeling.

The user can make the band around the selected isophote wider or narrower with a parameter to control the range. It delimits the zone where the new isophote can be redrawn. This band represents the area where the new shading can be affected by the redrawn isophote. Outside of this band, the shading will remain untouched. The user can redraw the isophote until he or she is satisfied.

4.2. Inflation

To inflate the mesh from its contour Γ_{contour} (Figure 4.4), we use the approach from Dvoroznak et al. [**9**] where the mesh is inflated using a linear system as described in Section 3.2.

As we take the square root of $\mathbf{z}$ from Equation 3.2.1, the height near Γ_{contour} tends to become jerky with an elevation that is too steep. We smooth the height $\mathbf{z}$ using the Weinkauf

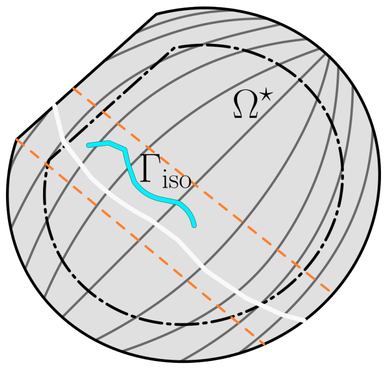


Figure 4.4 – The white line corresponds to the selected isophote. The cyan line corresponds to the redrawn isophote Γ_{iso}. The two orange dashed curves delimit the band. The dashed curves in black delimit region Ω^*. The grey curves are the strips computed from the known reflectance. The flat shading allows a better perception of the resulting underlying mesh.

method [**64**]

$$\min_z E_{\text{fairing}}(z) = \min_z \alpha E(z) + (1-\alpha)\int_\Omega ||z(\mathbf{x}) - z_0(\mathbf{x})||^2 d\mathbf{x} \tag{4.2.1}$$

where $z_0(\mathbf{x})$ represents the original height. The minimization can be discretized as

$$\min_{\mathbf{z}} \mathbf{z}^T(\alpha\mathbf{E} + (1-\alpha)\mathbf{M})\mathbf{z} - ((1-\alpha)\mathbf{z_0}^T\mathbf{M})\mathbf{z} \tag{4.2.2}$$

using the Galerkin method where $E(z)$ is assumed to be quadratic. Equation 4.2.2 can be solved through convex optimization. α is one of the parameters that the user can use to tweak inflation and influences the smoothness of the resulting $\mathbf{z}$. We found that $\alpha = 2\times 10^{-5}$ produces satisfying results.

Recently, Stein et al. [**58**] presented energy that can be used for smoothing

$$\frac{1}{2}\int_\Omega ||\nabla z||^2 dA \quad \text{Dirichlet} \tag{4.2.3}$$

$$\frac{1}{2}\int_\Omega (\Delta z)^2 dA \quad \text{Squared Laplacian} \tag{4.2.4}$$

$$\frac{1}{2}\int_\Omega ||\mathbf{H}(z)||_F^2 dA \quad \text{Hessian.} \tag{4.2.5}$$

Using the Galerkin method, each energy can be discretized as

$$\mathbf{zLz} \quad \text{Dirichlet} \tag{4.2.6}$$

$$\mathbf{zLM^{-1}Lz} \quad \text{Squared Laplacian} \tag{4.2.7}$$

$$\mathbf{zG^TAD\tilde{M}^{-1}D^TAGz} \quad \text{Hessian} \tag{4.2.8}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{G}$ is the gradient, $\mathbf{A}$ is the diagonal matrix of the area of each triangle, $\mathbf{D}$ is the divergence matrix, and $\tilde{\mathbf{M}}^{-\mathbf{1}}$ is a matrix where the inverse of the mass matrix is repeated four times along its diagonal [1]. Because the Hessian energy is only useful if the boundary conditions are not specified, we chose the squared Laplacian to smooth out the height.

4.3. Bounded Band and Impacted Region

In our method, the shading is altered locally. This makes it unnecessary to compute the height of vertices that are not affected when redrawing an isophote. Confining the region reduces computations and inaccuracies that can arise from our method. We start by identifying the bounded band where we interpolate the shading using the new isophote and the impacted region that will restrict the area where we integrate the strips and reconstruct the final mesh.

The strips from the method of characteristics are dependent on the gradient of the reflectance R and of the intensity E. This observation suggests that they can be used to pinpoint the area affected by the new isophote.

In our specific case where the light can only lies in the plane, we can deduce that every strip entering the contour of the mesh must leave it. Indeed, no normals for an implicit shape $z(x,y)$ can be perfectly aligned with the light direction which indicates that no strips should end at a peak of intensity. We can measure the flux traversing a domain with the following contour integral

$$\oint_{\partial\Omega} -\nabla_{p,q} R(p(t),q(t)) \cdot \mathbf{n}(t)\, dt \tag{4.3.1}$$

where R is the reflectance and $\mathbf{n}$ is the normal along the curve in the plane.

Using Green's theorem, we can transform this integral into

$$-\iint_{\Omega} \nabla_{x,y} \cdot \nabla_{p,q} R(p(x,y), q(x,y))\, dA \tag{4.3.2}$$

The amount of flux entering and quitting the domain cancels out (Equation 4.3.1) $\nabla_{x,y} \cdot \nabla_{p,q} R = 0$. Every strip thus enters and leaves the domain Ω.

1. More information can be retrieved on the different matrices in the annex of the original paper of Stein et al. [**58**]

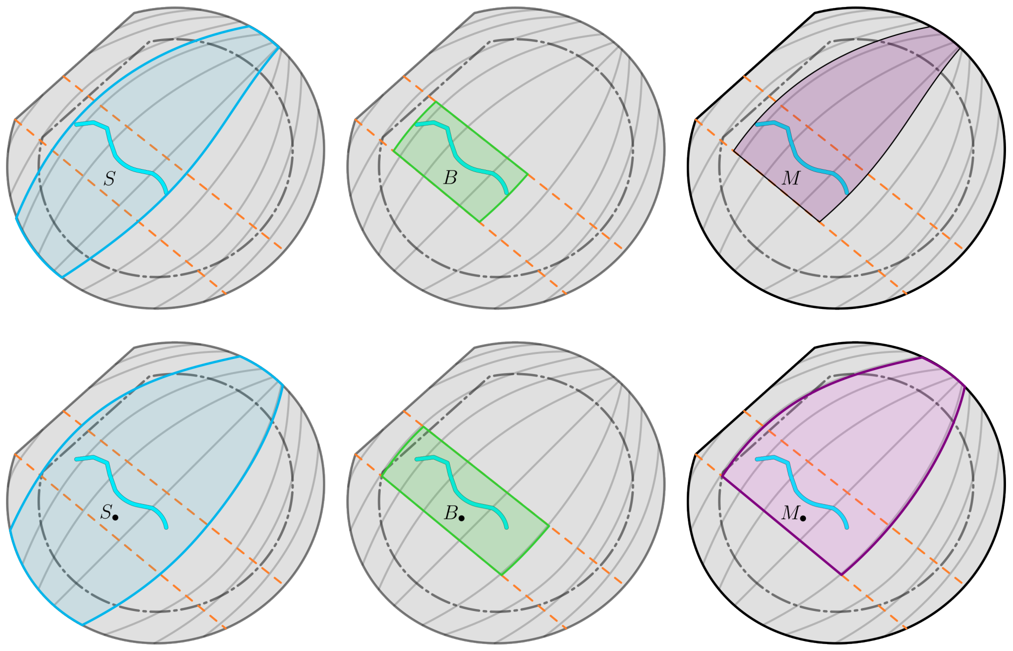


Figure 4.5 – On the left, regions S and $S_\bullet$ in blue represent the area covered by the strips intersecting the isophote Γ_{iso}. In the center, regions B and $B_\bullet$ in green represent the bounded band. On the right, regions M and $M_\bullet$ in purple represent the impacted area. At the top, each colored region represents the exact region while at the bottom, they are the approximated regions enclosed by the sampled strips.

The strips will traverse through two different points at the boundary, when they enter and leave the domain. However, knowing that Equation 3.1.1 is up to a constant, it is ambiguous which point should be used as an initial value. Indeed, characteristics strips can be integrated from the opposite direction which implies that two different initial conditions can be used. Without further constraint and assumption on the boundary, we take the one that would reach a peak of intensity if the light was not in the plane. We estimate that having the strips end at the boundary of the domain gives better visually results. Further investigation is needed to determine how we could use both initial conditions.

More formally, the method of characteristics needs to have an admissible initial curve (recall Equation 3.1.5). If we discard the points that are not admissible, multiple disconnected curves can give different results. When the shading is from a well-formed mesh, all initial curves should give the same result on the region where they overlap. However, if the shading is not from a well-formed mesh, we cannot guarantee that it will converge to the same outcome. We found that starting from the curves where $-\nabla\mathbf{R} \cdot \mathbf{n} > 0$ produces compelling

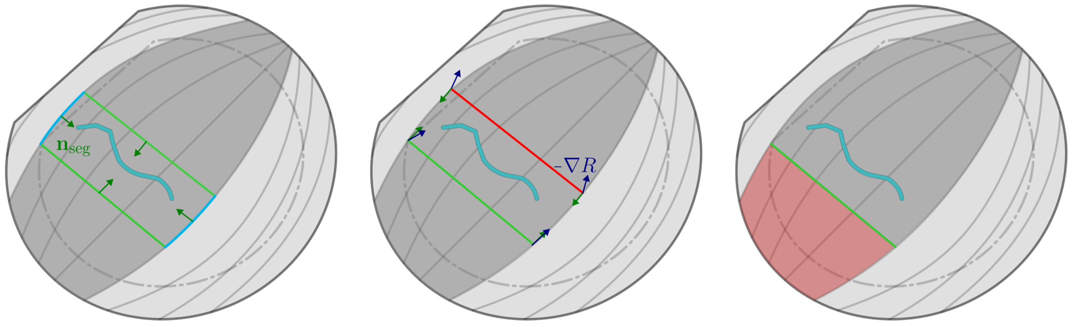


Figure 4.6 – Each figure illustrates a step to approximate M. The region in gray corresponds to $S_\bullet$. On the left, in green, we highlight $\partial B_\bullet$, and in blue, the part of $\partial B_\bullet$ that is common to $S_\bullet$. The normal of each segment $\mathbf{n}_{\text{seg}}$ is displayed in green. On the center, the two curves result from discarding $\partial B_\bullet \cap \partial S_\bullet$. The curve in red is discarded because the dot product is negative. On the right, we fragment $S_\bullet$ and discard the region in red because Γ_{iso} does not lie inside.

results that are intuitively and coherent with the changes in shading, but further investigation would be needed.

When a section of a strip traverses a region where the intensity E has been modified, its height will likely be modified too. The rest of the strip is not valid anymore and must be recomputed. Since we want to identify the region that is impacted by the change in shading, we compute the strips with the following ODE system by starting from the contour of the mesh

$$\begin{aligned} \frac{dx}{ds} &= -R_p(p,q) \\ \frac{dy}{ds} &= -R_q(p,q) \end{aligned} \tag{4.3.3}$$

where p and q are known at any given point on the surface.

For each point of the strips that do not intersect exactly with a vertex of a triangle, the values are computed using barycentric interpolation from values at the vertices. The strips from Equation 4.3.3 should never overlap because otherwise, Equation 3.1.2 would have multiple solutions, which is a contradiction.

The ODE can be solved efficiently using a simple Euler integration scheme. We found that more advanced integration schemes did not significantly improve the results, but did have an impact on performance. For this reason alone, we stick with a simple integration scheme.

In our method, the band area corresponds to the Minkowski sum between the selected isophote and a circle of radius equal to the range value in the panel. The band represents the area where the user can redraw the new isophote. The strips from the method of characteristics intersecting the new isophote form region S (Figure 4.5). The band area that is common to S corresponds to the new region B (Figure 4.5). To find the impacted region M (Figure 4.5), we consider the part of the strips that are affected after traversing region B. Small variations near $\partial\Omega$ can greatly impact the overall quality due to the steep increase in height over a small distance in the image plane. To reduce the impact of the small variations in height, we discard a small portion of the area outside of $\Omega^{\star}$. We use the superscript $\star$ to identify those reduced regions.

In practice, we compute a finite number of strips. We consider that strips in the region between sampled adjacent strips follow similar paths. We find which sub-region the new isophote intersects and use it to approximate region S. Since the new isophote is continuous, those subregions with adjacent non-intersect strips are merged into one fully connected region $S_{\bullet}$.

Instead of using the exact region S, we use the common area of $S_{\bullet}$ with the band to determine the approximated region $B_{\bullet}$. We refer to $B_{\bullet}^{\star}$ as the bounded band. To find the approximated impacted region $M_{\bullet}$ (Figure 4.6), we know that it will be a subregion of $S_{\bullet}$ and will traverse $\partial B_{\bullet}$. It is convenient to represent $B_{\bullet}$ by its contour. The contour is discretized through the ordered counterclockwise segments so that each normal segment (here tangent to the surface) points inward. To reduce unnecessary computations, we do not consider the parts of $\partial B_{\bullet}$ that do not overlap with $\partial S_{\bullet}$ and $\partial\Omega$. Since sampled strips are used as delimiters, they do not traverse $B_{\bullet}$ by definition. The process should result in two curves. By taking the extremity of the remaining curves, we discard the ones where $-\nabla R \cdot \mathbf{n}_{\text{seg}} < 0$ between the direction of the strips and segment normal. It implies that a strip traverses $B_{\bullet}$. The curve is used to fragment $S_{\bullet}$. $M_{\bullet}$ is the region where Γ_{iso} lies.

4.4. Localized Shading Interpolation

The method of characteristics requires the gradient of the intensity to be properly defined over the whole region of integration. Because the user redraws only a portion of the isophote, we interpolate locally the intensity inside the bounded band. We triangulate the area into a mesh by imposing that the redrawn isophote vertices from the polyline are all part of the

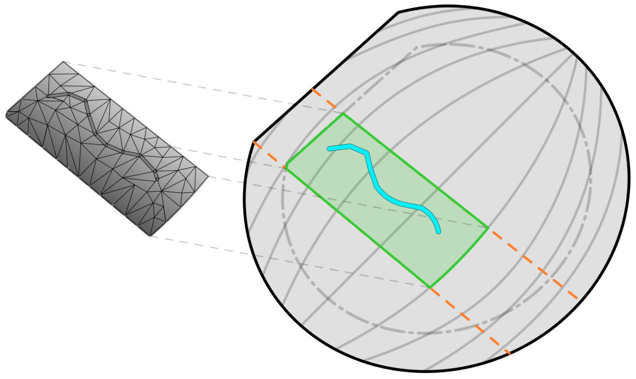


Figure 4.7 – By triangulating the inner region of the bounded band $B^{\star}_{\bullet}$ in green, we interpolate the value through the boundary by fixing the intensity at the vertices at the boundary $\partial B^{\star}_{\bullet}$ and at the vertices under isophote Γ_{iso}. The triangulated area illustrates the shading in grey after interpolation.

mesh. Then, we fix the intensity at the boundary $\partial B^{\star}_{\bullet}$ and the vertices issued from the redrawn isophote. We interpolate the intensity in the triangulated mesh inside of $B^{\star}_{\bullet}$. We compute the intensity $\mathbf{e}$ of the remaining vertices from the distinct mesh using

$$\begin{aligned} &\min_{\mathbf{e}} \mathbf{e}^T \mathbf{L}\mathbf{M}^{-1}\mathbf{L}\mathbf{e} \\ &\text{s.t. } \mathbf{e}_{\Gamma_{\text{iso}}} = c_{\text{iso}} \\ &\qquad \mathbf{e}_{\partial B} = \mathbf{g}_{\partial B} \end{aligned} \tag{4.4.1}$$

where c_{iso} is the intensity of the selected isophote, and $\mathbf{g}_{\partial B}$ corresponds to the intensity at the boundary found using barycentric interpolation from the original mesh intensity (Figure 4.7).

When integrating the strips, we sample the shading intensity $\mathbf{e}$ from the bounded band mesh when the strips pass through it. Otherwise, we sample the shading from the original mesh.

4.5. Strip Integration

Now that we have identified the impacted region and interpolated the shading in the bounded band area, we are ready to integrate along the strips using the same ODE system as Horn [**20**] used (Equation 3.1.2).

To integrate our system, we need to provide initial conditions. Because we only want to recompute the height in the impacted region $M_{\bullet}$, we start from the boundary ∂M where the strips cross (Figure 4.8). In practice, we can use the curve that we used to separate $S_{\bullet}$ to form $M_{\bullet}$ as explained in Section 4.3 and the part of $\partial\Omega$ that overlaps $\partial S_{\bullet}$. The remaining

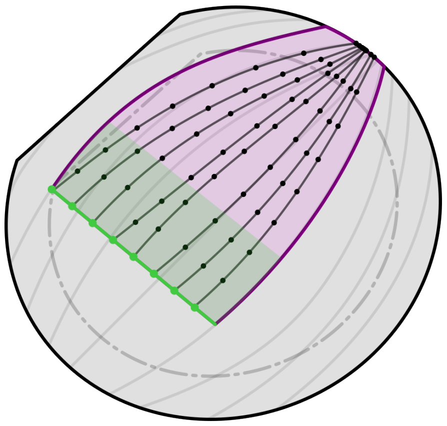


Figure 4.8 – Each green dot is a point used as an initial condition when its strip is integrated. The newly computed strips are in black. Each strip has been computed through a numerical solver. The black dots represent positions at different integration steps.

initial conditions p, q, and z are computed using barycentric interpolation from the original mesh. To ensure that the strips cover the impacted region, we sample at regular intervals on the section of $\partial M_\bullet$ where the strips cross.

In the original formulation of SFS, the boundary normals will lie in the plane because the boundary is occluding the rest of the mesh. However, this formulation is problematic for p,q coordinates because they cannot be used to represent normals that lie in the plane. Instead, the boundary normals are interpolated from adjacent triangle normals. Each triangle normal will never be perpendicular to the viewpoint.

To solve numerically the ODE system, we end up using an A-L stable [2]implicit solver from Kvaerno [**29**] with an accuracy of 5th order and with an adaptive step size of 4th order. The paths of strips are terminated when they reach the peak of intensity or when they exit region $M_\bullet$, which can arise if numerical errors accumulate through integration.

4.6. Shape Reconstruction

At this point in the process, the strips have been integrated, and we want to compute the height for the distinct mesh based on the newly computed strips.

Berger et al. [**3**] present a very comprehensive survey on a large array of reconstruction techniques. In our work, we focus on the radial basis function (RBF) method for its flexibility and general adoption. RBF can also be used beyond reconstruction for interpolating data.

2. More on the concept of stability for solving ODE in the book of Haiter et al. [**18**]

The RBF method builds a basis around a set of support points to interpolate a value at query points. In our method, the set of support points is composed of the first two coordinates from the integration steps for solving the ODE system (black dots in Figure 4.8) The first two coordinates of the vertices outside of $M_\bullet$ are seamlessly integrated into the set. The third coordinates from the integration step and the vertices outside of $M_\bullet$ are the height to be interpolated. The query points are the first two coordinates of all of the vertices of the original mesh.

The basis can be found by solving the following linear system

$$(\mathbf{K} + \lambda_{\text{smooth}} I)\mathbf{a} = \mathbf{z} \tag{4.6.1}$$

where λ_{smooth} is a smoothness factor, $\mathbf{a}$ is a weighted coefficient for each support point, and $\mathbf{z}$ is the height at the corresponding support points. Note that $\mathbf{z}$ and $\mathbf{a}$ are in bold to indicate a vector of values or points. Each entry of the kernel matrix $\mathbf{K}$ computes the radial basis $\phi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} \subseteq \mathbb{R}^2$ and $\mathbf{y} \subseteq \mathbb{R}^2$, between two points from the set of support points.

Possible choices for the radial basis function $\phi(\mathbf{x}, \mathbf{y})$ can be

$$\phi_{\text{m}}(\mathbf{x}, \mathbf{y}) = -\sqrt{1 + ||\mathbf{x} - \mathbf{y}||_2^2} \quad \text{Multiquadratic} \tag{4.6.2}$$

$$\phi_{\text{g}}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{y}||_2^2}{2\sigma}\right) \quad \text{Gaussian.} \tag{4.6.3}$$

To interpolate height at a set of query points, we retrieve it with

$$z(\mathbf{q}) = \mathbf{K}^*\mathbf{a} \tag{4.6.4}$$

where $\mathbf{K}^*$ is from the set of query points and the set of support points.

In our case, we found that $\lambda_{\text{smooth}} = 0.05$ and a multi-quadratic kernel produces decent results.

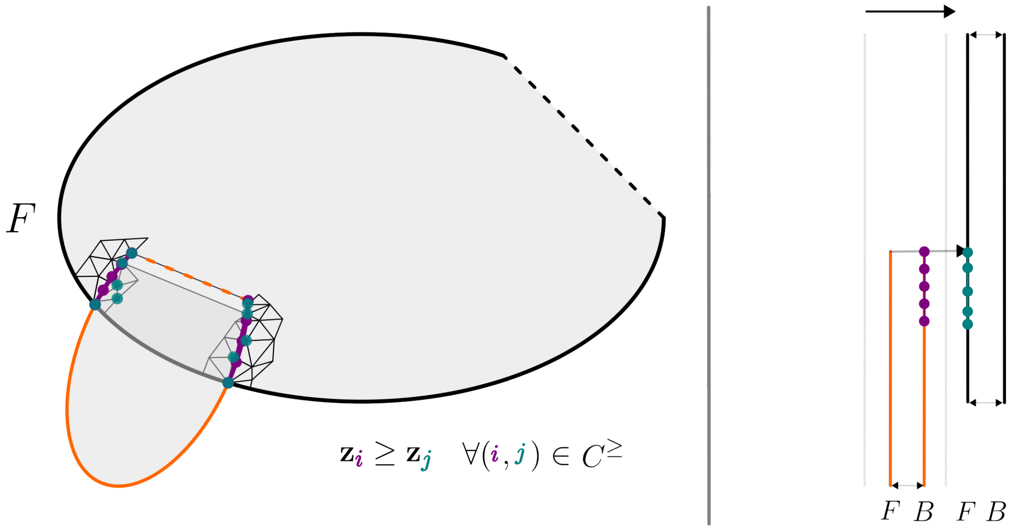
4.7. Assembling Layers

At this stage, the user has created and edited a number of mesh layers and is ready to assemble all mesh layers into a single mesh. We expand on the algorithm of Dvoroznak et al. [**10**] from *Monster Mash* to assemble each layer. In their work, Dvoroznak et al. [**10**] assemble each layer through a single pass where they consider at the same time the front face and its back face. However, we observed that we can process the front and the back faces into two separate passes and stitch them further along by preserving the height of each mesh instead of inflating it. Afterward, we bridge each front face and its back face together at their contour Γ_{contour}. Then, the vertices of each layer are shifted by using different inequality constraints to reduce interpenetration. We take special care to ensure that the approach of Dvoroznak et al. [**10**] is compatible with our method.

We start with the top of all front faces. We detect if a part of Γ_{conn} at the current face intersects one of the front faces below. We also make sure that the same part Γ_{conn} does not intersect two different faces below. We repeat the procedure for the next faces. We end up with pairs of faces to be stitched together (f_u^i, f_l^i) at Γ_{conn}^i where u is for upper, l is for lower, and i refers to the current pair. For each pair, we make a hole in the face f_l^i where Γ_{conn}^i is intersecting. Otherwise, we would end up with a non-manifold mesh. The creation of the hole introduces new vertices in the layer f_l^i. As the new vertices need to be assigned a height, it is computed from the original mesh using an RBF interpolation at the new vertices. Then, we stitch the vertices of f_u^i with f_l^i that lay at Γ_{conn}^i. For the back faces, we use the same procedure. Indeed, it is equivalent to stitching them with a viewpoint from the opposite side. The top of the front faces becomes the bottom of the back faces.

At this stage, we have connected front and back faces at Γ_{conn} respectively. Each vertex for the front and the back faces at Γ_{contour} share vertices at the same position in the image plane xy by symmetry. We then trivially bridge and triangulate the front and back faces together at their boundary. To make it aesthetically more pleasing, the front face of each layer is displaced above its back face by a small margin, making the vertices at Γ_{contour} non-self-intersecting. This stage is nearly identical to the approach of Dvoroznak et al. [**10**].

Instead of using the same inequality constraint as Dvoroznak et al. [**10**] as explained in Section 3.2, we found that we could reduce the number of constraints while keeping the same visual outlook on the shape. In their work, $C^{\geq}$ is the set of vertices at Γ_{contour} of the front face which is on top of the vertices of the front face below. Instead, we rearrange it to be the vertices at Γ_{contour} of the back face being on top of the nearest vertices of the front face below. Similarly, $C^{\leq}$ becomes the set of vertices at Γ_{contour} of the front face behind the nearest vertices of the back face above. We found that the constraint set $C^{=}$ is unnecessary for our method (Figure 4.9). For each set, the nearest vertices can be found using a standard K-nearest neighbors algorithm on the projected vertices in the image plane. Similarly to Dvoroznak et al. [**10**], we could have reused their approach to animate the mesh.



F
$\mathbf{z}_i \geq \mathbf{z}_j \quad \forall(i, j) \in C^{\geq}$
F B F B

Figure 4.9 – On the left, the dots illustrate the constraint on the vertices that are part of $C^{\geq}$. $C^{\geq}$ corresponds to the set of constraints where the z components of the orange back face are above the z components of the black front face. For a vertex on the contour of the orange face, we pick the nearest vertex in the face below. On the right, the same dots illustrate how we impose the constraints on the orange back face and the black front face.

Chapter 5

Results

In this chapter, we provide an overview of our results. We present several cases to illustrate how different parameters affect the results and compare our results to those of *Monster Mash* from Dvoroznak et al. [**10**]. Instead of rewriting basic operations, we relied on several libraries. We implemented our system using Jax [**7**] for accelerating computations on GPU, and Diffrax [**26**] for providing ODE solvers based on Jax. Libigl [**23**] and Gpytoolbox [**55**] simplified geometry processing tasks. Shapely [**15**] was used for boolean processing and basic geometric queries in 2D. ModernGL and SDL were used to render our application and create an interactive experience. Cairo allowed us to draw polylines into a texture. ImGUI helped to create the interface.

As demonstrated in Figure 5.1, we investigate the effects of modifying isophotes at two different isovalues. Our findings reveal that for an isophote $R = 0$, the decrease in height occurs more slowly compared to an isophote $R > 0$, as both cases illustrate. Interestingly, even when changes in shading appear subtle, as seen in the last row and columns of Figure 5.1, the corresponding changes in height can be quite pronounced. In some instances, these changes are so significant that they cause strips to dip below the xy plane. To reduce such artifacts, we cap the strip height to 0 when the value is negative, which can explain the dark region at the top right corner of the "pillow" and "m" meshes. This phenomenon can be explained by the inaccuracy arising from our integration scheme and the fact that the newly interpolated shading is not guaranteed to be associated with a shape that has its boundary lying in the image plane. Another difference lies in the size of the impacted region. A strip will invariably intersect the $R = 0$ isophote before it reaches an isophote $R > 0$. Consequently, the impacted region for a redrawn isophote at $R = 0$ is going to

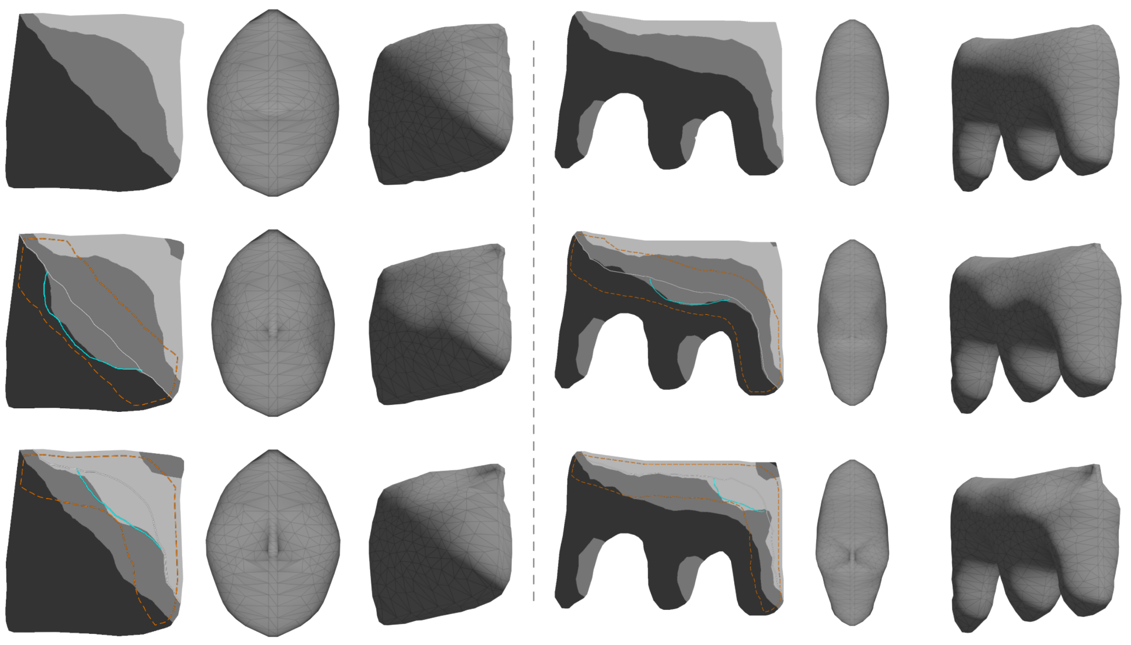


Figure 5.1 – On the left, we show our method on a convex "pillow" shape. On the right, we show our method on a concave "m" shape. The rows show the original shading for the mesh (top) and after editing for isophote $R = 0$ (middle), and isophote $R > 0$ (bottom). The column at the center of both shapes shows the meshes from the light direction. The column at the right shows the meshes from a different viewpoint. For each experiment, we used a radius of 0.3 and a light direction $\mathbf{l} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}^T$. Isophotes $R > 0$ are harder to distinguish. To make it easier, we display the shading using a cell shading with three tones.

be broader. When dealing with a redrawn isophote $R > 0$, fewer steps are necessary for a numerical solver compared to an isophote at $R = 0$. It implies that numerical errors have less opportunity to accumulate in the case of isophote $R > 0$, potentially leading to more accurate and reliable outcomes. However, we found that it is more intuitive to modify the isophote at $R = 0$ due to deformation being easier to predict. As observed in the perceptual study of Xu et al. [**67**], redrawing accurately isophotes can be challenging. Most artists can reproduce how isophotes behave, but struggle to draw them for specific isovalues, which is needed to produce precise and desired results with our method.

In Figure 5.2, we illustrate the impact that varying the range of the band has on the reconstructed height. Increasing the range of the band extends the area over which shading is interpolated. The interpolation, however, does not account for the curvature of the mesh,

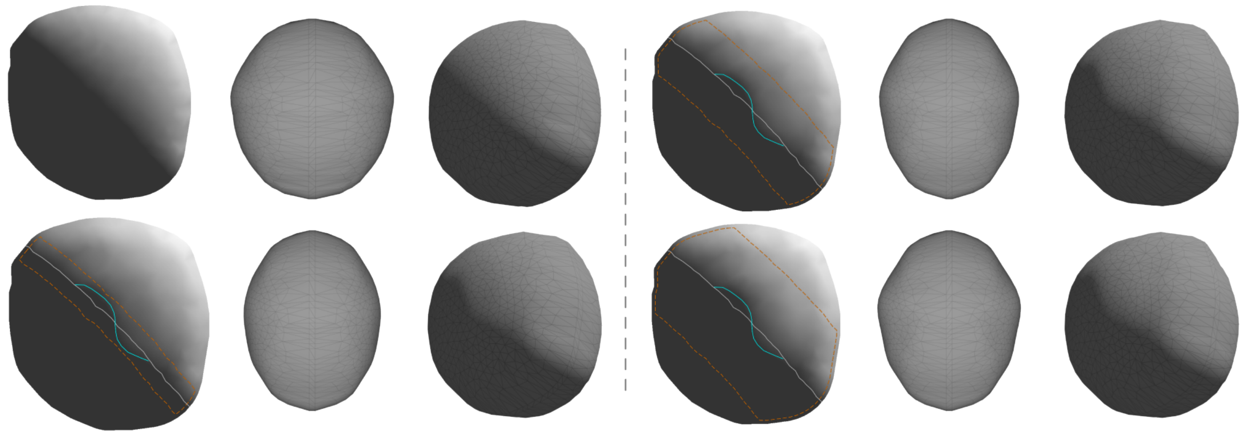


Figure 5.2 – The shaded columns depict the shading from reference mesh before (top left) and after being edited in our system. Each shape has been edited with the same new isophote with different radiuses for the bounded band: 0.2 (bottom left), 0.5 (top right), 0.8 (bottom right). The remaining columns provide an alternative view of the mesh associated with the shaded columns. For each of them, the light is set at $\mathbf{l} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}^T$.

leading to potential distortions. High values for the band, even if the shading does not appear to be visually altered much, can have a non-intuitive impact on the shape. Having the boundary condition set tighter around the isophote implies that we can make the area of interpolation smaller and neglect those distortions, but the change in height is less visually perceptible. Furthermore, predicting the behavior of the strips near the mesh boundary is challenging. Extending the radius of the band to the boundary of partial Ω^* can result in unexpected modifications in height in certain areas of the mesh, emphasizing the complex relation between shading and height.

Figure 5.3 demonstrates the influence of the mesh resolution on the quality of our solution. The second column of this figure highlights how varying resolutions impact the shading. The number of vertices, and therefore of query points, is larger, which makes reconstructing the height from strips smoother. The shading becomes smoother too. A higher resolution leads to a larger number of triangles in the mesh, which in turn increases the computational demands of our algorithm. This is because our algorithm's complexity is directly linked to the number of triangles in the mesh. On the upside, a high-resolution mesh with more triangles allows for more accurate barycentric interpolation between vertices. This finer interpolation reduces bias in our solution, resulting in a more accurate and visually appealing representation.

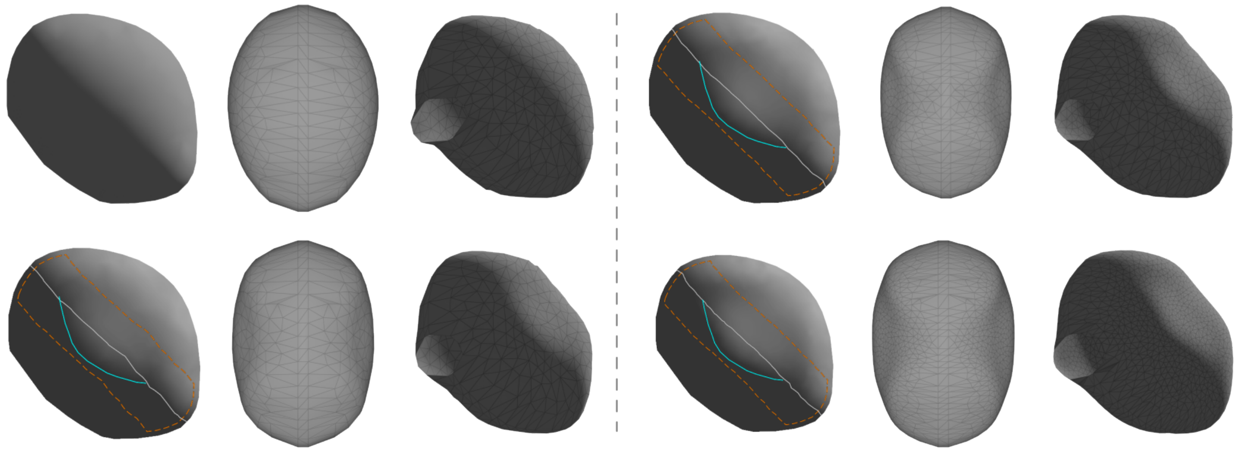


Figure 5.3 – The shaded columns depict the shading from reference mesh before (top left) and after being edited in our system. For each row except the top left, we present the results for low (210 vertices and 375 triangles), medium (389 vertices and 723 triangles), and high (782 vertices and 1488 triangles) resolutions. The remaining columns provide an alternative view of the mesh associated with the shaded columns. For each of them, the light is set at $\mathbf{l} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}^T$, we used the same new isophote and a radius of 0.3.

In Figure 5.4, we showcase the application of multiple iterations of our method to intricately edit a shape and modify its height profile. Our process involves interpolating shading based on the boundary of the bounded band $\partial B^{\star}_{\bullet}$. However, this can inadvertently eliminate finer details, emphasizing the need for careful consideration of the range used in each iteration. As we progress through each iteration, the shading complexity of the shape increases, as depicted by the white line on each shape. This increase in complexity is directly linked to the evolving complexity of the overall distributions of height on the shape. A notable example is iteration e) applied to a convex shape, where the isophote becomes less apparent, making it challenging to discern. This complexity in shading interpretation can lead to non-trivial adjustments, and in some cases, unintended modifications. This is particularly evident in iterations j), k), and l) on the concave shape, where these complexities result in alterations that might not align with the intended outcome. Indeed, complex height profiles create complex shading, which in turn makes it difficult to intuitively understand how the shape will be altered. The shading at iterations e) and f) can illustrate how such phenomena can arise. The white isophote does not behave intuitively, which makes the bounded band much larger than one would have wanted. The resulting characteristic strips of such shading can reach different regions of the shape, which can result in unexpected behavior.

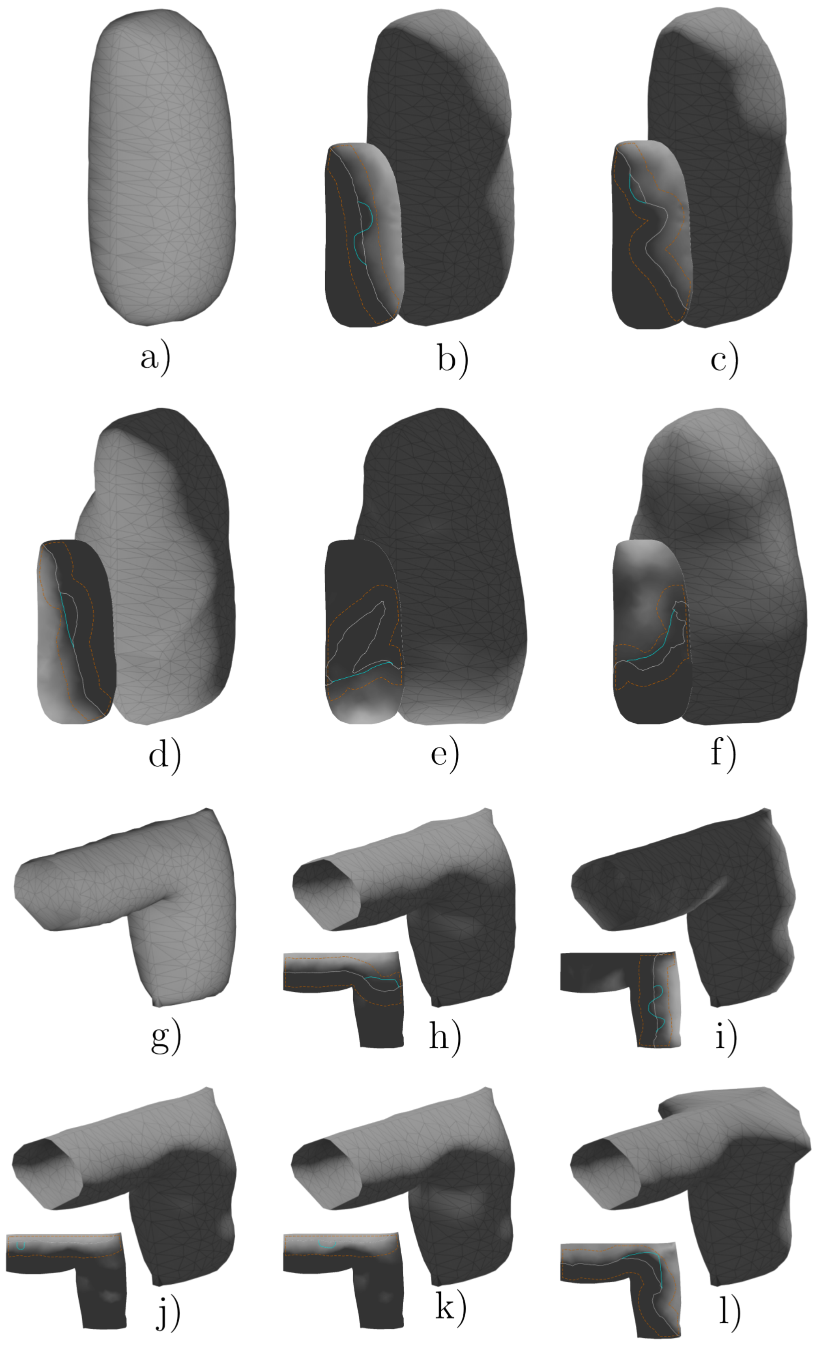


Figure 5.4 – We showcase two distinct series of edits on two types of shapes: convex (top) and concave (bottom). Each series is labeled as rows a sequential iteration of our method. In the bottom-left corner of each sub-figure, we include a drawing that illustrates the specific modifications (modified isophote) made to the shading at that particular stage of our editing process.

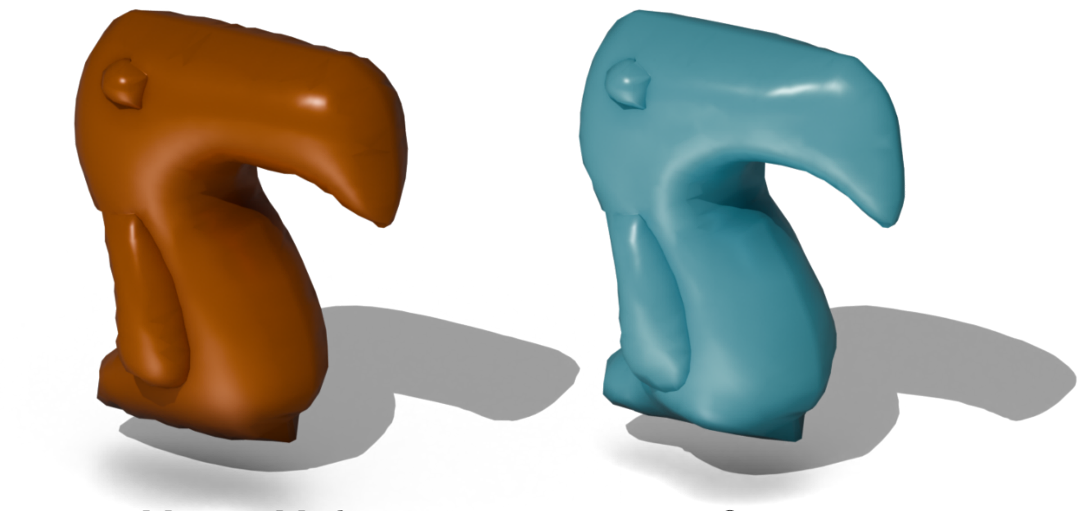


Figure 5.5 – In orange, the result from *Monster Mash* by Dvoroznak et al. [**10**]. In blue, the result of our method after multiple iterations. Our method provides additional control compared to *Monster Mash* [**10**]. The specular reflection is added to improve shape perception, but it is not part of our method.

Comparing our method to *Monster Mash* from Dvoroznak et al. [**10**], Figure 5.5 illustrates the enhanced capability for a user to create more refined height variations. Using our technique, the user was able to achieve several modifications on the toucan-like mesh. These modifications include slimming down the beak, enlarging the belly, slightly carving a tail, and flattening the wing. Each of these alterations contributes to a more convincing and nuanced shape. This comparison highlights the advantages of our method over Dvoroznak et al. [**10**] proposition, especially in terms of the level of control it offers. However, even if it is used successfully, the tool can be challenging for casual 3D modeling. Altering a shape is not straightforward due to the underlying complexity of shading.

Our method specifically restricts users from altering shading for light sources not lying in the drawing plane, as doing so is not robust and can negatively impact the quality of the results. Reconstructing height from various light directions significantly affects the outcome. In Figures 5.6 and 5.7, the height was reconstructed for the whole region by starting the

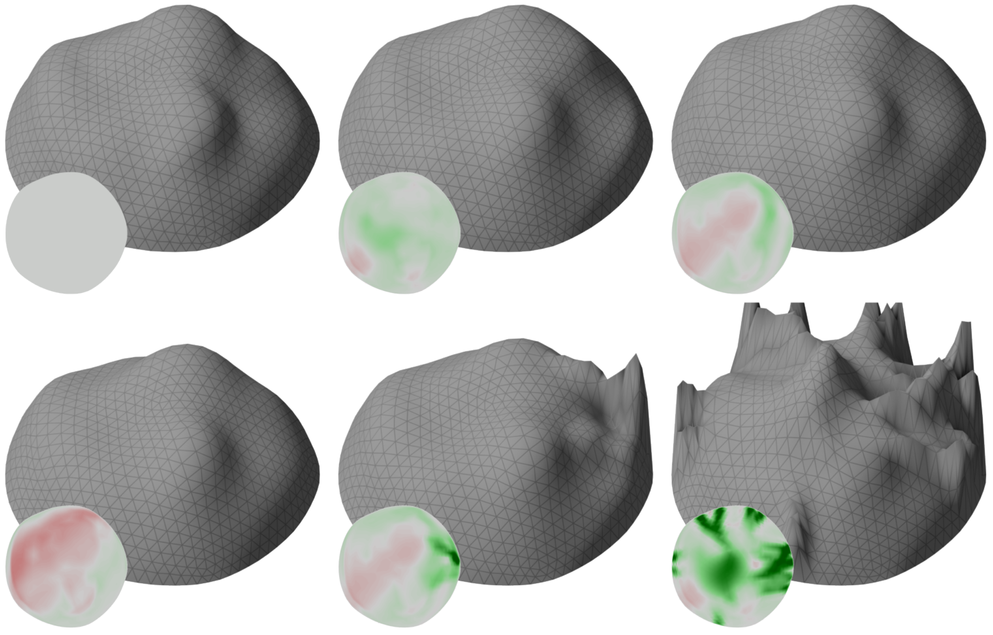


Figure 5.6 – Reconstruction of height utilizing the method of characteristics, initiated from the boundary of the shape. The top left image exhibits the reference to be reconstructed. Each column represents the light source direction $\mathbf{l}$: $\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\end{bmatrix}^T$, $\begin{bmatrix}\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}}\end{bmatrix}^T$, and $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. The strips from the method of characteristics are trimmed in the top row and untrimmed in the bottom row. An inset at the bottom left corner of each image displays a color-coded height map based on the variation of height compared to the reference mesh.

integration from the boundary. A key observation from the last row for each shape is that as strips approach a peak in the geometric model, they begin to diverge in height, leading to spikes in the reconstructed mesh. This divergence can be mitigated by trimming the strips that cross over each other in the image plane, a technique that we have used to enhance results as shown in the first row of each series. We discard the strips that diverge above the ones below in their z coordinates. Notably, when the light source is in the plane, we avoid trimming the strips. Although they may cross in the image plane, this does not substantially impair height reconstruction, and trimming in such cases would excessively prolong computation time.

In addition, using an alternative formulation for Lambertian reflectance, we rewrote the problem as

$$F(x,y,p,q) = \sqrt{(1+p^2+q^2)}E(x,y) - (-l_x p - l_y q + l_z) \tag{5.0.1}$$

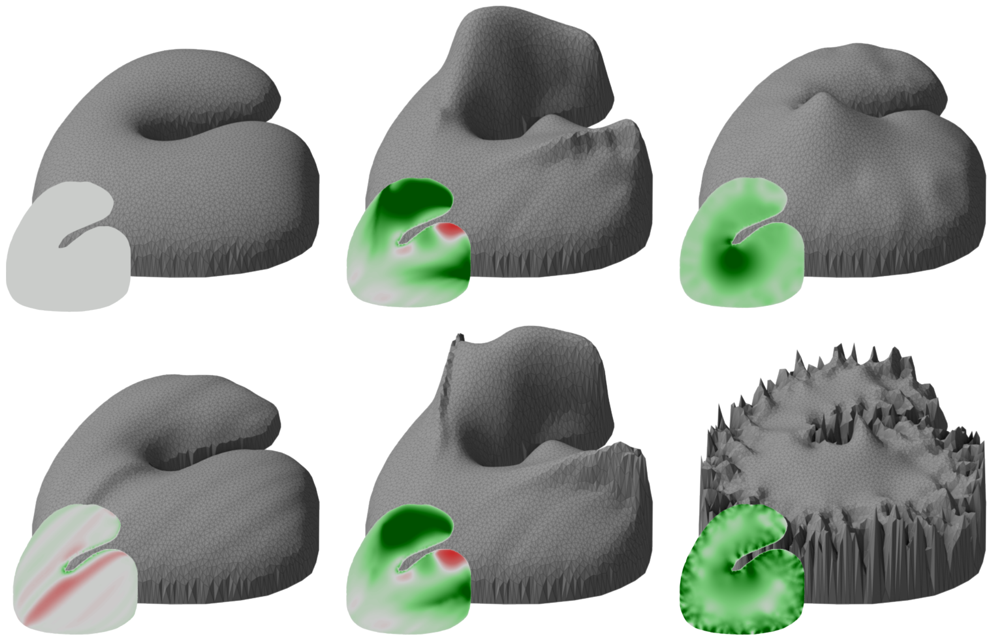


Figure 5.7 – Same configuration than in Figure 5.6, but for a concave shape with a higher resolution.

which led to the following system of equations using the method of characteristics

$$
\begin{aligned}
\frac{dx}{ds} &= \frac{p}{\sqrt{1+p^2+q^2}}E + l_x \\
\frac{dy}{ds} &= \frac{q}{\sqrt{1+p^2+q^2}}E + l_y \\
\frac{dp}{ds} &= -\sqrt{1+p^2+q^2}\frac{dE}{dx} \\
\frac{dq}{ds} &= -\sqrt{1+p^2+q^2}\frac{dE}{dy} \\
\frac{dz}{ds} &= p\left(\frac{p}{\sqrt{1+p^2+q^2}}E + l_x\right) + q\left(\frac{q}{\sqrt{1+p^2+q^2}}E + l_y\right).
\end{aligned}
\tag{5.0.2}
$$

As we can see, strips can be integrated without explicit dependence on l_z. We believe that the divergence of our solution, especially when the strips get closer to a peak, is a result of the integration being influenced by both shading and mesh geometry, and not directly linked to the inclination of the light. Interestingly, when applying the same equations to implicit shapes outside of our system, where shading at each point is precisely defined as opposed to a triangle mesh, the method performs better. It suggests that the smoothness of our surface plays a major role in the integration process. We did not observe a particular difference between Equations 3.1.2 and 5.0.2. We kept the latter because the formulation is more general.

Chapter 6

Conclusion

In our work, we have explored a method for artists to edit meshes using a non-traditional approach. Traditional tools for 3D creation have a steeper learning curve before being effectively used. We offer a method that relies on sketching to indirectly edit the height of a mesh using methods developed for SFS. We employ the method of characteristics from Horn [**20**] to identify affected regions, and recompute heights from shading using characteristic strips. We elaborate on the mathematical and algorithmic framework that allows our interactive system to sketch isophotes on meshes created and inflated from their contours. Meshes from different layers are assembled into one pleasant-looking mesh based on the framework of Dvornozak et al. [**10**]. We show and discuss results created with our method.

Our method presents a few limitations. We cannot use a light source direction that does not lie in the plane of the drawing canvas, which makes our method less flexible and less intuitive. If the number of characteristic strips does not cover the domain uniformly when we try to identify various regions, which can arise for complex shading, it can result in regions that are larger than they should be. To retrieve the height from the characteristic strips, we interpolate the height from the sampled integration step into each vertices of the mesh being edited using RBF interpolation. The interpolation can fail to interpolate the height properly on the existing mesh because of a lack of points or of too much space between each strip due to the fall-off associated with multi-quadratic kernels. Interpolating the shading in the bounded band offers no guarantee that the newly interpolated shading will produce a shape that respects that the boundary lies in a plane. Lastly, by keeping one specific orientation for our characteristic strips, we limit the part of the boundary that can be used as an initial curve instead of using the complete boundary. Although using isophotes is an intriguing approach to editing shading, it is not as straightforward as we were hoping due to the complexity of shading. Altering the shading can result in non-intuitive behavior. Future research is necessary to alleviate such current limitations. We will elaborate on multiple avenues to extend our method.

Future Work. In addressing the challenges of Shape-From-Shading (SFS), we propose exploring alternative solutions beyond traditional reliance on characteristics. A promising approach involves solving the Hamiltonian-Jacobian formulation of SFS. This can be achieved through an iterative process, transforming our problem into an eikonal equation by fixing parameters. Recently, there has been a focus on the potential of machine learning in various fields, which could be utilized in SFS. We have observed that the reconstruction of height in SFS is significantly impacted by the smoothness of underlying shading, a factor that may prove insufficient under certain lighting conditions. Exploring neural ODE [**8**] in this context offers an interesting prospect. To effectively train the neural network, one could utilize strips derived from Equation 4.3.3 as a training dataset. The training dataset could be from scanned deformable surfaces where each surface would have a different isophote for a known shape and light direction.

Beyond SFS, there have been notable advancements in Generative AI, addressing similar challenges. However, these developments often rely on inputs based on real-life scenery, contrasting with the stroke-based inputs typical in sketching. This discrepancy raises the issue of dataset reliability, as sketching encompasses a wide range of styles. Despite these challenges, investigating the application of Generative AI techniques in sketch-based SFS is a compelling avenue for future research.

Our current method for SFS supports the alteration of one isophote at a time. An extension of our work could handle multiple changes simultaneously. Presently, our approach is limited to creating meshes by inflating their contours. One could expand this method to work with arbitrary meshes rather than just altering heights of meshes from a plane. Unfortunately, occlusion remains a major issue to overcome. We could address this by altering the shading within the intrinsic space of the mesh where the strips would be perfectly continuous. Implementing UV parameterization could be a viable solution for achieving this task.

Moreover, our method has so far been applied only to Lambertian reflectance. Accommodating more complex reflectance models, including both physical and non-physical BRDFs, represents another avenue. Sketching, in particular, often employs non-photorealistic reflectance models. Investigating a broader spectrum of reflectance models and exploring how our method could be adapted to go beyond Lambertian reflectance would be an interesting area of research.

We neglected how real artists perceive changes in shading as a shape changes. It would be crucial to have an in-depth user study of all the implications of getting real artists in the loop and providing closer measurements and expectations in various shape-lighting-shading configurations.

References

[1] Jonathan T. Barron and Jitendra Malik. Shape, Illumination, and Reflectance from Shading, October 2020. arXiv:2010.03592 [cs].

[2] Harry Barrow, J Tenenbaum, A Hanson, and E Riseman. Recovering intrinsic scene characteristics. *Comput. vis. syst*, 2(3-26):2, 1978.

[3] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, 36(1):301–329, January 2017.

[4] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198, San Jose California, July 1977. ACM.

[5] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. *Polygon Mesh Processing*. A K Peters/CRC Press, Natick, Mass, 1st edition edition, October 2010.

[6] Frederic Boudon, Christophe Pradal, Thomas Cokelaer, Przemyslaw Prusinkiewicz, and Christophe Godin. L-Py: An L-System Simulation Framework for Modeling Plant Architecture Development Based on a Dynamic Language. *Frontiers in Plant Science*, 3, 2012.

[7] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[8] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations, December 2019. arXiv:1806.07366 [cs, stat].

[9] Marek Dvorožňák, Saman Sepehri Nejad, Ondřej Jamriška, Alec Jacobson, Ladislav Kavan, and Daniel Sýkora. Seamless reconstruction of part-based high-relief models from hand-drawn images. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pages 1–9, Victoria British Columbia Canada, August 2018. ACM.

[10] Marek Dvorožňák, Daniel Sýkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. Monster mash: a single-view approach to casual 3D modeling and animation. *ACM Transactions on Graphics*, 39(6):1–12, December 2020.

[11] Ady Ecker and Allan D. Jepson. Polynomial shape from shading. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 145–152, San Francisco, CA, USA, June 2010. IEEE.

[12] Lawrence C. Evans. *Partial differential equations.* Number v. 19 in Graduate studies in mathematics. American Mathematical Society, Providence, R.I, 2nd ed edition, 2010. OCLC: ocn465190110.

[13] Filippo Andrea Fanni, Fabio Pellacini, Riccardo Scateni, and Andrea Giachetti. PAVEL: Decorative Patterns with Packed Volumetric Elements. *ACM Transactions on Graphics*, 41(2):1–15, April 2022.

[14] R.T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439–451, July 1988.

[15] Sean Gillies, Casper van der Wel, Joris Van den Bossche, Mike W. Taves, Joshua Arnott, Brendan C. Ward, and Others. Shapely, October 2023.

[16] Yotam Gingold and Denis Zorin. Shading-based surface editing. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 1–9, New York, NY, USA, 2008. Association for Computing Machinery.

[17] J. Hadamard. Sur les problèmes aux dérivés partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52, 1902. tex.citeulike-article-id: 7133163.

[18] E. (Ernst) Hairer and Gerhard. Wanner. *Solving Ordinary Differential Equations II : Stiff and Differential-Algebraic Problems.* Springer Series in Computational Mathematics, 0179-3632 ; 14. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

[19] Berthold K. P Horn and Michael J Brooks. The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208, February 1986.

[20] Berthold KP Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. 1970.

[21] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 409–416, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[22] Katsushi Ikeuchi and Berthold K. P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17(1):141–184, August 1981.

[23] Alec Jacobson, Daniele Panozzo, and others. libigl: A simple C++ geometry processing library, 2018.

[24] James T. Kajiya. The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20(4):143–150, 1986.

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023.

[26] Patrick Kidger. On Neural Differential Equations, February 2022. arXiv:2202.02435 [cs, math, stat].

[27] Ron Kimmel and Alfred M. Bruckstein. Tracking Level Sets by Level Sets: A Method for Solving the Shape from Shading Problem. *Computer Vision and Image Understanding*, 62(1):47–58, July 1995.

[28] Ron Kimmel and James A. Sethian. Optimal Algorithm for Shape from Shading and Path Planning. *Journal of Mathematical Imaging and Vision*, 14(3):237–244, May 2001.

[29] A. Kværnø. Singly Diagonally Implicit Runge–Kutta Methods with an Explicit First Stage. *BIT Numerical Mathematics*, 44(3):489–502, August 2004.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Number: 7553 Publisher: Nature Publishing Group.

[31] Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. BendSketch: modeling freeform surfaces through 2D sketching. *ACM Transactions on Graphics*, 36(4):125:1–125:14, 2017.

[32] Aristid Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, March 1968.

[33] P. L. Lions, E. Rouy, and A. Tourin. Shape-from-shading, viscosity solutions and edges. *Numerische Mathematik*, 64(1):323–353, December 1993.

[34] Claudio Mancinelli, Giacomo Nazzaro, Fabio Pellacini, and Enrico Puppo. b/Surf: Interactive Bézier Splines on Surface Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 29(7):3419–3435, July 2023.

[35] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

[36] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In Gerald Farin, Hans-Christian Hege, David Hoffman, Christopher R. Johnson, Konrad Polthier, Hans-Christian Hege, and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. Series Title: Mathematics and Visualization.

[37] Giacomo Nazzaro, Enrico Puppo, and Fabio Pellacini. *geoTangle* : Interactive Design of Geodesic Tangle Patterns on Surfaces. *ACM Transactions on Graphics*, 41(2):1–17, April 2022.

[38] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics*, 40(6):1–13, December 2021.

[39] J. Oliensis. Existence and uniqueness in shape from shading. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume i, pages 341–345, Atlantic City, NJ, USA, 1990. IEEE Comput. Soc. Press.

[40] J. Oliensis. Shape from shading as a partially well-constrained problem. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 559–564, Maui, HI, USA, 1991. IEEE Comput. Sco. Press.

[41] J. Oliensis. Uniqueness in shape from shading. *International Journal of Computer Vision*, 6(2):75–104, June 1991.

[42] Michael Oren and Shree K. Nayar. Generalization of Lambert's reflectance model. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*, pages 239–246, Not Known, 1994. ACM Press.

[43] Alex P. Pentland. Local Shading Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(2):170–187, March 1984. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[44] Alex P. Pentland. Linear shape from shading. *International Journal of Computer Vision*, 4(2):153–162, March 1990.

[45] Ken Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, July 1985.

[46] Tsai Ping-Sing and Mubarak Shah. Shape from shading using linear approximation. *Image and Vision Computing*, 12(8):487–498, October 1994.

[47] Ulrich Pinkall and Konrad Polthier. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics*, 2(1):15–36, January 1993.

[48] E. Prados and O. Faugeras. Shape from Shading: A Well-Posed Problem? In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 870–877, San Diego, CA, USA, 2005. IEEE.

[49] Emmanuel Prados and Olivier Faugeras. A Generic and Provably Convergent Shape-from-Shading Method for Orthographic and Pinhole Cameras. *International Journal of Computer Vision*, 65(1):97–125, November 2005.

[50] Emmanuel Prados, Olivier Faugeras, and Elisabeth Rouy. Shape from Shading and Viscosity Solutions. In Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision — ECCV 2002*, volume 2351, pages 790–804. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. Series Title: Lecture Notes in Computer Science.

[51] V. S. Ramachandran. Perception of shape from shading. *Nature*, 331(6152):163–166, January 1988. Number: 6152 Publisher: Nature Publishing Group.

[52] Marzia Riso, Giacomo Nazzaro, Enrico Puppo, Alec Jacobson, Qingnan Zhou, and Fabio Pellacini. BoolSurf: Boolean Operations on Surfaces. *ACM Transactions on Graphics*, 41(6):1–13, December 2022.

[53] Elisabeth Rouy and Agnès Tourin. A Viscosity Solutions Approach to Shape-From-Shading. *SIAM Journal on Numerical Analysis*, 29(3):867–884, 1992. Publisher: Society for Industrial and Applied Mathematics.

[54] Hiroaki Santo, Masaki Samejima, and Yasuyuki Matsushita. Numerical shape-from-shading revisited. *IPSJ Transactions on Computer Vision and Applications*, 10(1):8, June 2018.

[55] Silvia Sellán, Oded Stein, and others. gptyoolbox: A python geometry processing toolbox, 2023.

[56] J. A. Sethian. Fast Marching Methods. *SIAM Review*, 41(2):199–235, January 1999. Publisher: Society for Industrial and Applied Mathematics.

[57] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, SGP '07, pages 109–116, Goslar, DEU, 2007. Eurographics Association.

[58] Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. Natural Boundary Conditions for Smoothing in Geometry Processing. *ACM Transactions on Graphics*, 37(2):1–13, April 2018.

[59] Thomas M. Strat. A Numerical Method for Shape-From-Shading From A Single Image. Working Paper, MIT Artificial Intelligence Laboratory, January 1979. Accepted: 2008-04-02T14:50:55Z.

[60] Daniel Sýkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transactions on Graphics*, 33(2):1–15, March 2014.

[61] Johan Wagemans, Andrea Doorn, and Jan Koenderink. The Shading Cue in Context. *i-Perception*, 1:159–78, December 2010.

[62] Yu Wang and Justin Solomon. Intrinsic and extrinsic operators for shape analysis. In *Handbook of Numerical Analysis*, volume 20, pages 41–115. Elsevier, 2019.

[63] Gregory J. Ward. Measuring and modeling anisotropic reflection. *ACM SIGGRAPH Computer Graphics*, 26(2):265–272, 1992.

[64] Tino Weinkauf, Yotam Gingold, and Olga Sorkine. Topology-based smoothing of 2D scalar fields with C1-continuity. In *Proceedings of the 12th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'10, pages 1221–1230, Chichester, GBR, 2010. The Eurographs Association & John Wiley & Sons, Ltd.

[65] Ying Xiong, Ayan Chakrabarti, Ronen Basri, Steven J. Gortler, David W. Jacobs, and Todd Zickler. From Shading to Local Shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):67–79, January 2015.

[66] Q. Xu, Y. Gingold, and K. Singh. Inverse toon shading: interactive normal field modeling with isophotes. In *Proceedings of the workshop on Sketch-Based Interfaces and Modeling*, SBIM '15, pages 15–25, Goslar, DEU, 2015. Eurographics Association.

[67] Qiuying Xu, Songrun Liu, Yotam Gingold, and Karan Singh. Using isophotes and shadows to interactively model normal and height fields. *Computers & Graphics*, 59:130–142, October 2016.

[68] Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. Physics-based differentiable rendering: from theory to implementation. In *ACM SIGGRAPH 2020 Courses*, SIGGRAPH '20, pages 1–30, New York, NY, USA, 2020. Association for Computing Machinery.