Computational Methods of Optimization and Geometry Processing for Physics Applications

by

Jacob Spainhour

B.S., Florida State University, 2020

M.S., University of Colorado Boulder, 2023

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2025

Committee Members:

Stephen Becker, Chair

Nick Bottenus

Eduardo Corona

Adrianna Gillman

Kenneth Weiss

Spainhour, Jacob (Ph.D., Applied Mathematics)

Computational Methods of Optimization and Geometry Processing for Physics Applications

Thesis directed by Prof. Stephen Becker

This thesis presents novel computational methods in two distinct, yet not disjoint subfields of applied mathematics, and is consequently divided into two parts.

In Part I, we consider methods of numerical optimization as applied to problems in scientific experimentation, in which a limited quantity of available resources must be carefully allocated to maximize the performance of the experiment. In the first chapter of Part I, this takes the form of discovering optimized ultrasound transmission sequences which, when imaged in the REFoCUS framework, produce high quality images. In the second chapter of Part I, this takes the form of discovering optimized measurement protocols for the verification of quantum states through fidelity estimation.

In Part II, we consider methods of computational geometry as applied to problems in multiphysics simulation, in which geometrically complex objects must be computationally handled in a way that is reliable and robust to numerical imperfection. In the first and second chapter of Part II, we develop methods of evaluating the generalized winding number for general curves and surfaces, from which we derive methods of determining containment for geometric objects which otherwise do not define an interior volume. In the third chapter of Part II, we develop a method of 3D material interface reconstruction by means of optimally defining a collection of half-spaces whose convex intersection has geometric moments as close as possible to provided reference data.

Dedication

To my dad, whose kindness and strength of character I hope to reflect in my own life. And to my mom, the original Dr. Spainhour, for being an unending source of pride and inspiration.

Acknowledgements

I have limitless gratitude for the many different communities that I've had the privilege to be a part of, and the support networks that I've relied on over the course of my PhD career.

In no particular order:

I'm very grateful to my friends and colleagues in APPM, and in particular to my advisor Stephen Becker for trusting me with the freedom to pursue my own, perhaps unconventional research trajectory. I'm also grateful to my co-advisor Nick Bottenus for showing me the joys of working with outside collaborators, a mindset that has come to define my Ph.D. career.

I'm particularly indebted to the Bottenus lab for the chance to cross paths with Nazli, whose love and support has undoubtedly kept my last few years as a student not just bearable, but vibrant and exciting.

I'm very grateful to my internship mentors Kenneth Weiss and Mikhail Shashkov for imbuing in me the skills of a career mathematician. Indeed, I am uniquely grateful to Kenny for taking that first chance on me, which has since become the most pivotal opportunity of my academic career.

And finally, I'm grateful to my friends and family back in Tallahassee, to which a part of me will always belong. Without the support of my parents, my siblings, and the childhood friends that I'm lucky to still have in my life, I simply wouldn't be the person I am today.

All in all, I'm just happy to have so many people who have positively impacted my life that my final moments of a PhD student will be spent whittling my gratitude down to a single page of 11pt text. You all have turned this into quite the challenge, and I really appreciate it.

Contents

Tables

Table

Figures

Figure

Introduction

The defining feature of a career in applied mathematics is the opportunity to work on problems in an incredibly broad collection of application areas. This thesis serves as a clear demonstration of this idea, and represents the culmination of a number of different research projects in a number of different problem domains. For clarity of exposition in this document, we distinguish the projects presented (largely retroactively) according to the field of applied mathematics which is most relevant to the presented solution.

In Part I, we consider work in which the primary contribution relates to **Numerical Optimization**, and involves applications in ultrasound imaging systems and quantum state verification.

In Part II, we consider work in which the primary contribution relates to **Computational Geometry**, and involves applications in computer graphics and computational multiphysics.

In total, the principal contributions of this thesis are as follows, in order of presentation:

(1) We create a custom machine learning architecture that generates an optimized sequence of ultrasound transmissions that, when used with the REFoCUS imaging framework, produces image quality superior to conventional alternatives. The relevant machine learning model leverages a novel differentiable beamforming apparatus that makes training on a large and entirely simulated dataset computationally feasible.

(2) We describe a strategy for optimizing the distribution of quantum measurements for the purpose of estimating the fidelity between an experimental quantum state and its intended target. These optimized measurement allocations are capable of generating a better worst-

case error estimate than conventional alternatives, while also suggesting a reasonable importance weighting among measurements.

(3) We define a numerically stable algorithm to accurately evaluate the generalized winding number for unstructured collections of rational parametric curves in 2D and trimmed NURBS surfaces in 3D. In turn, these algorithms permit evaluation of robust and consistent containment classifications with respect to non-watertight and self-intersecting shapes for which conventional methods cannot apply.

(4) We develop a 3D moment-of-fluid method for interface reconstruction that represents recovered geometry as the convex intersection of multiple half-spaces, optimizing their parameters to minimize the least-squares error between computed and reference geometric moments. The stability of this non-linear optimization procedure is improved using a novel normalization procedure, enabling reliable convergence and accurate reconstruction of complex geometric features.

Part I: Numerical Optimization for Scientific Experiments

Chapter 1

Introduction to Part I

The fundamental mathematical framework for optimization is concerned with the discovery of the minimum value for an objective function f given input x, subject to the constraint that $x \in C$, or,

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in C, \end{aligned} \tag{1.1}$$

Despite this relatively simple framing, in truth one can encode a truly staggering number of problems through careful selection of f and C. In particular, we are concerned with problems that apply the principles of numerical optimization to scenarios in the physical sciences where a careful allocation of experimental resources is necessary, namely **Array Encoding for Ultrasound Imaging** and **Measurement Allocation for Quantum Fidelity Estimation**.

In the first problem, we consider the transmit encoding model for ultrasound imaging, in which sound waves are emitted into tissue according to a pre-defined transmission sequence, and the backscattered echoes are collected and processed into an image through the REFoCUS imaging framework. Naturally, careful selection of the transmission sequence is critical for ensuring that the resulting image is viable for consideration by a clinician or as part of some other image processing pipeline, but as our previous work has explored, there exist many competing objectives that influence their design [20]. Furthermore, it is often necessary to altogether restrict the number of transmissions used, as the acquisition of data is quite time-consuming relative to what is practical

in a clinical setting.

In an effort to find the best possible transmission sequence for a given imaging scenario, we solve Equation 1.1 with f as an objective measure of image quality across all possible ultrasound data for given transmission sequence x, provided that $x \in C$, the search space of sequences which utilize as few experimental resources as feasible. To make the concept of "all possible ultrasound data" computationally tractable, we treat this minimization problem through a machine learning framework, which permits optimization on successive subsets of representative training data.

Even still, our choice to use a loss function which is based directly on image quality would ordinary incur an impractical computational cost and memory footprint for even reasonably sized batches of training data. In our work, we compensate for this through the use of a custom formula for the analytic gradient of our imagine process, which when implemented directly into the automatic differentiation capabilities of PyTorch [128], allows rapid evaluation of backpropagation steps during training. Differentiable imaging techniques of this kind are becoming popular in the ultrasound literature [95], notably in our own work studying calibration and motion tracking for swept synthetic aperture imaging modalities [143].

Furthermore, our proposed machine learning model features a set of trainable parameters which exactly correspond to physical hardware settings of the ultrasound device. This "interpretable" approach to a machine learning framework offers a number of practical advantages over conventional deep learning strategies, most notably a simple inability to produce hallucinations in the imaging process. Specifically, we only learn the set of hardware parameters through our machine learning framework, and once this experimental setup has been computed, the actual imaging is performed via classical techniques. In total, the contents of Chapter 2 is based on the work **Optimization of Array Encoding for Ultrasound Imaging**, published in Physics in Medicine and Biology in June 2024 [167].

In the second problem, we consider the verification of contemporary quantum devices. Any implementation of such a device comes at an incredible economic cost, and the currently unavoidable imperfections in their construction mean that extra resources must be spent just to ensure that

the quantum state created matches expectation. We consider the problem through the lens of fidelity, a measure of closeness between the expected and experimental quantum states that is largely compatible with many conventional methods of statistical analysis [86]. At the same time, contemporary techniques for fidelity estimation often assume a particular type of measurement system [50], and so the relevant analysis becomes entirely unfounded when applied to even common types of quantum measurement systems.

In order to find the best allocation of a more flexible collection of available measurements, we solve Equation 1.1 by taking f to be the worst-case error estimate for a given allocation x, where C defines a total budget that must be distributed among measurements. Importantly, this condition that f measures the "worst-case" error means that even evaluating f for a single experimental protocol involves solving a *separate* optimization problem, whose solution method is introduced in our prior work [152]. To solve this inner level of optimization as efficiently as possible, we reframe the problem statement in a way that is compatible with contemporary convex optimization software, namely `CVX`. By leveraging the exceptional performance of these libraries, even the outer level of optimization can be solved at a reasonable cost.

In both subproblems presented in Part I, the shared high-level objective is the discovery of underlying patterns in seemingly unstructured or otherwise difficult to parse data. For encoded ultrasound imaging, it is clear from prior work that the relationship between data acquisition and image quality was often far from straight-forward, and our machine learning approach was able to find acquisition sequences that outperformed a wide variety of conventional alternatives despite having no discernible structure. Similarly in the context of quantum fidelity estimation, existing theory could only reliably predict the optimal measurement sequence in a very limited experimental context, while our optimization approach not only finds an optimal allocation of more versatile experimental resources, but can even order the optimal usage of these resources in a way that is more aligned with available hardware.

Chapter 2

Optimization of Array Encoding for Ultrasound Imaging

2.1 Abstract

The transmit encoding model for synthetic aperture imaging is a robust and flexible framework for understanding the effects of acoustic transmission on ultrasound image reconstruction. Our objective is to use machine learning (ML) to construct scanning sequences, parameterized by time delays and apodization weights, that produce high-quality B-mode images. We use a custom ML model in PyTorch with simulated RF data from Field II to probe the space of possible encoding sequences for those that minimize a loss function that describes image quality. This approach is made computationally feasible by a novel formulation of the derivative for delay-and-sum beamforming. When trained for a specified experimental setting (imaging domain, hardware restrictions, etc.), our ML model produces optimized encoding sequences that, when deployed in the REFoCUS imaging framework, improve a number of standard quality metrics over conventional sequences including resolution, field of view, and contrast. We demonstrate this experimentally on both wire targets and a tissue-mimicking phantom. The results of this chapter demonstrate that the set of commonly used encoding schemes represent only a narrow subset of those available. Additionally, they demonstrate the value for ML tasks in synthetic transmit aperture imaging to consider the beamformer within the model, instead of purely as a post-processing step.

In collaboration with Korben Smart, Stephen Becker, and Nick Bottenus

2.2 Introduction

In classical ultrasound imaging, transmissions from a transducer array are focused towards specific locations in the target medium, and the backscattered echoes are collected and processed into an image that is clear at these locations [34]. More modern systems use a synthetic transmit aperture system that combines the echoes from multiple transmissions to recreate this focus across multiple points in the region. In the ideal case, data acquisition is performed by firing each array element individually in sequence and echoes are received by every element in parallel. The full set of element-to-element signals is then combined into a B-mode image via post-processing that achieves focus even at depth [84]. However, the large number of transmissions needed paired with the relatively low power of each means that, when performed directly, this procedure creates images with poor frame rate and low signal-to-noise ratio (SNR), making it impractical for clinical application [28].

Retrospective Encoding For Conventional Ultrasound Sequences, or REFoCUS, is an alternative imaging framework that considers the responses from an *arbitrary* transmission as a linear combination of these pairwise element responses [19, 3]. In this way, the transmission sequence described by time delays and apodization weights parameterizes an *encoding* of the ground-truth basis of element-to-element signals, the multistatic (STA) data set. By using the specific sequence used to acquire data, the echoed response signals can be *decoded* to produce a robust approximation of this basis. This approximation can then be focused throughout the imaging domain as post-processing, making a B-mode image that achieves the desired SNR and focal depth. Conventional transmission sequences are often selected according to geometric principles (e.g., focused, planewave, or diverging beams) or according to spatial codes (e.g., Hadamard [28] or S-sequence [67]). However, the REFoCUS framework generalizes the encoding framweork to allow for a uniform treatment of other categories of transmissions [20].

The quality of the resultant B-mode image—measured in terms of resolution, field of view (FOV), and artifacts—is highly dependent on properties of the transmit sequence used for data

acquisition. As an example of the importance of beam geometry, focused beams provide very high resolution, but only around a particular focal point. Another influential property is simply the number of transmissions. While the number of transmissions ideally matches or exceeds the number of array elements, practical frame rate considerations often restrict this. In the resulting underdetermined case, there is active research that seeks to minimize the loss of information during encoding and decoding. Some groups have shown success using randomly assigned delays, as such schemes lead to a very well conditioned encoding, in the sense that the relevant encoding matrices are full rank and therefore stably invertible [184]. Similarly, a spatial code for element apodizations can be applied to produce a lossless encoding (or near-lossless when the matrix must be truncated [186]), which in turn increases SNR during imaging.

However, the space of all possible encoding sequences dwarfs those with these sorts of immediately clear and desirable properties [165]. In this chapter, we present a novel machine learning (ML) model for data acquisition and imaging which we train to efficiently search the set of all possible transmission sequences, identifying those that lead to high-quality images using the REFoCUS encoding framework. Notably, our training workflow is designed to measure the quality of a sequence after image formation, better reflecting how performance is judged in a clinical setting. Furthermore, the only trainable parameters of this machine learning model are a physical description of the transmission sequence, such that a trained ML model is parameterized by an optimized encoding sequence. This optimized encoding sequence can then be considered independently of the ML model that generated it, allowing it to be deployed classically within the REFoCUS framework to achieve image quality beyond what is attainable with conventional sequences.

2.2.1 Related Work

Techniques in machine learning have gained a significant amount of research attention in the field of ultrasound imaging, particularly in the application of deep neural networks (DNNs) [100, 76]. Following the unprecedented successes of the deep convolutional neural network (CNN) AlexNet in non-medical imaging tasks [92], similar networks are frequently trained and utilized for ultrasound-

adjacent tasks as an additional post-processing step to a conventional imaging pipeline by performing classification, segmentation, etc. on image data [115].

Other deep learning approaches perform image formation directly. In a prototypical setup, a DNN is fed the received echoes of a transducer array to produce a B-mode image, aiding or even supplanting the use of a conventional beamformer [177, 54]. Often the goal is to create a network that can be evaluated faster than classical imaging techniques without degrading image quality [26]. Other times, the DNN is designed to directly *improve* image quality. For example, [75] utilize a CNN as a beamformer that reduces speckle, citing considerable advantages over conventional delay-and-sum techniques.

While a well-trained DNN can offer these and other benefits, their use necessarily introduces complications. Most notably, neural networks are susceptible to hallucinations, in which it unexpectedly generates features that are not present in the underlying data, an issue that is uniquely consequential in medical imaging [15]. Our approach to machine learning entirely circumvents this and other issues by using a custom architecture whose forward operation is entirely acoustically motivated. Specifically, the proposed ML method improves imaging quality *exclusively* through an optimized parameterization of experimental hardware via the transmit sequence. After data is acquired with an optimized sequence, the B-mode image to be evaluated is generated through the REFoCUS imaging framework, which itself is based solely on the linear nature pulse-echo responses (We describe the decoding and imaging processes in detail in Section 2.3.1 and Section 2.3.2 respectively).

A similar strategy to our own is employed by [26], in which a machine learning model is trained to optimize the apodization weights of planewave transmissions, although a DNN is used to decode the received echoes. Specifically, a stacked denoising autoencoder architecture (a type of DNN whose first trainable layer is taken to be the weights themselves) is trained to optimize the reconstruction of the multistatic data set. In contrast, our ML model allows full control of the encoding sequence through a continuous treatment of apodization weights *and* time delays, with no other trainable parameters. At the same time, our training workflow optimizes the encoding

sequence according to direct measures of image quality. In doing so, we provide notable counterexamples to the principle that a better reconstruction of the multistatic data set leads to higher quality images.

By detaching our machine learning model from conventional neural network architectures, our approach obtains theoretical and practical advantages beyond avoiding hallucinations. First, an optimized encoding sequence generated by our ML model can be analyzed separately from the model itself, such that its beneficial features can be identified and further studied. This is in contrast to a trained DNN, whose per-layer weights and biases have no meaning outside the context of the DNN. Second, the smaller parameter space of our ML model dramatically reduces the training time needed to generate an optimized encoding sequence. This allows for greater flexibility in the overall ML workflow, as a different configuration of the ML model can be retrained at a minimal cost when the experimental setting changes (e.g., targeting a different viewing depth or prioritizing different image features via an alternate loss function). Finally, optimization of data acquisition through the transmit sequence makes our imaging process very robust to different targets. While conventional DNNs are typically trained for very specific imaging tasks on a constrained set of training data, we will show that the proposed approach not only generalizes well to out-of-sample training data, but also to data with dramatically different features. This is because our ML model is fundamentally optimized based on the underlying acoustic principles of ultrasound imaging, where the processing of backscattered echoes is influenced primarily through manipulation of the transmission sequence.

2.3 Methods

2.3.1 Transmit Encoding Theory

We consider a linear transducer array of N_E physical elements, of which N_T are capable of transmitting a diverging wave and N_R can receive backscattered echoes. Moreover, it is typical to have elements that perform both functions, such that $N_T = N_R = N_E$. Taken across all pairs of transmit and receive elements, collected RF (radio frequency) signals form a time dependent

$N_T \times N_R$ matrix $\mathbf{U}(t)$, within which the matrix element $u_{TR}(t)$ is the signal observed by transmitting on element T and receiving on element R. In the synthetic transmit aperture model, the complete collection of pulse-echo response signals between pairs of transmit and receive elements make up the multistatic data set. Because these signals can be delayed and summed across the receive channel to produce transmit focus throughout the domain, as in delay-and-sum (DAS) beamforming, these data can be considered the mathematical basis of our imaging [84].

As previously discussed, it is inadvisable to construct $\mathbf{U}(t)$ directly by firing each transmit element in sequence across a total of N_T transmissions [28]. Instead, the REFoCUS technique seeks to construct an approximation of $\mathbf{U}(t)$ using the responses from an arbitrary scanning sequence. We consider a scanning sequence of N_M separate transmissions, each of which is parameterized by N_T time delays t_{MT} and apodization weights w_{MT}. The response observed by receive element R from transmit M can be described as a weighted sum of time delayed matrix elements from the multistatic data set,

$$s_{MR}(t) = \sum_{T=1}^{N_T} w_{MT} u_{TR}(t - t_{MT}) \,, \tag{2.1}$$

which are similarly collected into a time dependent $N_M \times N_R$ matrix $\mathbf{S}(t)$ of focused responses. Collected across each transmit, these time delays and apodization weights make up the *encoding sequence*, denoted $\boldsymbol{\sigma} = (t, w)_{MT}$.

It is now convenient to work in the frequency domain, where each time delay of $u_{TR}(t)$ becomes a complex phase shift of $u_{TR}(\omega)$ [3]. For simplicity, we represent the frequency dependent Fourier transform of $\mathbf{U}(t)$ and $\mathbf{S}(t)$ as $\mathbf{U}(\omega)$ and $\mathbf{S}(\omega)$ respectively. In doing so, we can express the transmission response of receive element R to transmission M as

$$s_{MR}(\omega) = \sum_{T=1}^{N_T} w_{MT} \exp(-j\omega t_{MT}) u_{TR}(\omega) \,. \tag{2.2}$$

This describes a linear relationship between the multistatic data set $\mathbf{U}(\omega)$ and the transmission responses $\mathbf{S}(\omega)$ that is fully parameterized by $\boldsymbol{\sigma}$, given by

$$\mathbf{S}(\omega) = \mathbf{H}(\omega)\mathbf{U}(\omega) \,, \tag{2.3}$$

where $\mathbf{H}(\omega)$ is a frequency dependent $N_M \times N_T$ *encoding matrix* given by

$$\mathbf{H}(\omega) = \begin{bmatrix} w_{1,1}\exp(-j\omega t_{1,1}) & \dots & w_{1,T}\exp(j\omega t_{1,T}) \\ w_{2,1}\exp(-j\omega t_{2,1}) & \dots & w_{2,T}\exp(j\omega t_{2,T}) \\ \vdots & \ddots & \vdots \\ w_{M,1}\exp(-j\omega t_{M,1}) & \dots & w_{M,T}\exp(j\omega t_{M,T}) \end{bmatrix}. \tag{2.4}$$

In practice, we collect only discrete, equispaced samples from $\mathbf{S}(t)$, from which we have a total of N_ω angular frequencies ω. Because calculations involving the collection of matrices $\mathbf{S}(\omega)$ are typically performed in parallel across frequencies, we use calligraphic letters to denote the collection of all frequencies. As an example, we denote $\boldsymbol{\mathcal{S}}_\omega = \mathbf{S}(\omega)$, and represent the above equation more compactly as $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{U}}$. When we wish to emphasize the dependence of our encoding matrices $\boldsymbol{\mathcal{H}}$ on the sequence $\boldsymbol{\sigma}$, we add $\boldsymbol{\sigma}$ as a subscript.

The objective of the REFoCUS framework is to create an approximation of the multistatic data set $\widehat{\boldsymbol{\mathcal{U}}}$ from the recorded echoes in $\boldsymbol{\mathcal{S}}$. This is done by applying a frequency dependent *decoder* $\mathbf{H}^\dagger(\omega)$ to $\mathbf{S}(\omega)$, resulting in

$$\widehat{\boldsymbol{\mathcal{U}}} = \boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{S}} = (\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}})\boldsymbol{\mathcal{U}}\,. \tag{2.5}$$

This decoder is theoretically arbitrary, with options explored in the literature ranging from a per-frequency conjugate transpose [19] to a fully trained neural network [26]. In this chapter, we follow [3, 20] and apply a Tikhonov regularized pseudoinverse that has been shown to work well experimentally. This choice of decoder depends only on the original encoding matrix, which is itself dependent only on our sequence $\boldsymbol{\sigma}$. The Tikhonov decoder is given explicitly by

$$\boldsymbol{\mathcal{H}}^\dagger = (\boldsymbol{\mathcal{H}}^*\boldsymbol{\mathcal{H}} + \gamma^2\boldsymbol{\mathcal{I}}_{N_E})^{-1}\boldsymbol{\mathcal{H}}^*\,, \tag{2.6}$$

where $\boldsymbol{\mathcal{H}}^*$ is the conjugate transpose of $\boldsymbol{\mathcal{H}}$ and γ is a regularization constant. This regularization both suppresses the contribution of noise to the recovered multistatic channel data and produces a more stable inversion of $\boldsymbol{\mathcal{H}}$, which together results in a more accurate and robust recovery of

$\widehat{\boldsymbol{\mathcal{U}}}$. This is particularly important in the underdetermined system of interest, in which the number of transmits N_M is significantly smaller than the number of array elements N_E. In all usages of regularization herein we take a value of $\gamma = 0.1\sigma_{\max}(\omega)$, where $\sigma_{\max}(\omega)$ is the spectral norm of each per-frequency encoding matrix $\mathbf{H}(\omega)$. This particular value has been observed empirically to avoid issues of over- or under-regularization in the presence of noise [20].

2.3.2 Beamforming and Imaging Procedure

Given a multistatic data set $\boldsymbol{\mathcal{U}}$, we produce the 2D B-mode image of interest with conventional DAS beamforming. This technique maps ground-truth or approximated multistatic IQ data (in-phase and quadrature data derived from the experimentally acquired RF data) at each frequency to a single IQ data matrix, taking a sum of the time delayed data in $\widehat{\boldsymbol{\mathcal{U}}}$. By manipulating these post-acquisition time delays, an image is created that is focused at individual pixels in the imaging domain. Mathematically, we denote this beamforming operation as $\mathcal{B} : \mathbb{C}^{N_M \times N_R \times N_\omega} \to \mathbb{C}^{N_x \times N_z}$, where (N_x, N_z) are the number of pixels in the lateral and axial directions of our image.

To make the resulting image suitable for interpretation by a human observer, we apply a number of non-linear post-processing steps to this re-focused IQ data. These include the computation of the signal envelope, log-scaling the dynamic range, and clipping the image to a minimum decibel value. In our notation, we represent these operations collectively as $|\cdot|_{\text{Im}} : \mathbb{C}^{N_x \times N_z} \to \mathbb{R}^{N_x \times N_z}$. When applied in the experimental context, the total forward operation of our ML model takes in transmission responses collected in $\boldsymbol{\mathcal{S}}$ and an encoding sequence $\boldsymbol{\sigma}$, and produces an image represented by the real matrix $|\mathcal{B}(\boldsymbol{\mathcal{H}}_{\boldsymbol{\sigma}}^{\dagger}\boldsymbol{\mathcal{S}})|_{\text{Im}}$.

To ensure that the generated encoding sequences is useful in a clinical setting, we train our ML model with a loss function $\mathcal{L} : \mathbb{R}^{N_x \times N_z} \to \mathbb{R}^{+}$ that directly measures image quality. While there is flexibility in the specific choice of loss function [74], achieving a lower value for the loss should correspond to higher resolution, higher SNR, wider FOV, fewer artifacts, etc.

A clear choice is to use a loss function that quantifies these qualities directly. For example, the generalized contrast to noise ratio (gCNR) is a measure of histogram overlap between the

brightness of predefined speckle and anechoic regions, such that a higher value indicates greater contrast in the image [142]. We can in principle improve the gCNR by minimizing the loss function

$$\mathcal{L}_{\mathrm{gCNR}}(\widehat{\mathbf{X}}) := 1 - \mathrm{gCNR}(\widehat{\mathbf{X}})\,, \tag{2.7}$$

However, complications with implementation make this loss function impractical. In particular, the histogram operator is not differentiable and standard continuous approximations are unstable, which makes the metric incompatible with backpropagation during training.

On the other hand, a natural and easily implementable choice is a normalized ℓ^2 comparison to a reference image that has desirable qualities. This introduces further flexibility in the choice of imaging target, which we discuss in Section 2.3.4 and Section 2.5.3. With a given image $\widehat{\mathbf{X}}$ and reference image $\mathbf{X}$, this loss is then defined by

$$\mathcal{L}_{\ell^2}(\widehat{\mathbf{X}}; \mathbf{X}) := \frac{1}{N_x N_z} \left\| \mathbf{X} - \widehat{\mathbf{X}} \right\|_2^2 . \tag{2.8}$$

Importantly, we will show that training the ML model to minimize $\mathcal{L}_{\ell^2}$ still makes a meaningful improvement to the gCNR of produced images.

2.3.3 Implementation of the Proposed Machine Learning Model

We define an optimized encoding sequence $\boldsymbol{\sigma}^*$ as one that, for a given loss function $\mathcal{L}$, beamformer $\mathcal{B}$, and decoder $\boldsymbol{\mathcal{H}}_{\boldsymbol{\sigma}}^{\dagger}$, minimizes the non-convex optimization problem

$$\min_{\boldsymbol{\sigma} \in \Sigma} \mathbb{E}\, \mathcal{L}\left(|\mathcal{B}(\boldsymbol{\mathcal{H}}_{\boldsymbol{\sigma}}^{\dagger} \boldsymbol{\mathcal{S}})|_{\mathrm{Im}} \right) , \tag{2.9}$$

where this expectation is taken over all possible transmission responses $\boldsymbol{\mathcal{S}}$. As it is impossible to evaluate this expectation directly, we instead approach the minimization problem from a machine learning perspective, where we instead minimize over a representative sample of training data $\{\boldsymbol{\mathcal{U}}_i\}$. Each piece of ground-truth multistatic data is related to a transmission response by our encoding matrix $\boldsymbol{\mathcal{H}}_{\boldsymbol{\sigma}}$ by $\boldsymbol{\mathcal{S}}_i = \boldsymbol{\mathcal{H}}_{\boldsymbol{\sigma}} \boldsymbol{\mathcal{U}}_i$.

Similarly, we must restrict the space of encoding sequences to those that are practically realizable on a physical system, which we denote as Σ. For example, we limit the total transmission

power by restricting the set of apodization weights $\{w_{MT}\}$ to an ℓ^∞ ball, effectively clipping each value to the range $[-1, 1]$. Other restrictions stem from quantization of numerical values to match machine clock cycles. These small discretizations are much more dependent on the specific acquisition system, but also less impactful to the optimized result.

Altogether, we are left to solve

$$\boldsymbol{\sigma}^* = \underset{\boldsymbol{\sigma}\in\Sigma}{\arg\min} \frac{1}{N_U} \sum_{i=1}^{N_U} \mathcal{L}\left(|\mathcal{B}(\boldsymbol{\mathcal{H}}_{\boldsymbol{\sigma}}^\dagger \boldsymbol{\mathcal{H}}_{\boldsymbol{\sigma}} \boldsymbol{\mathcal{U}}_i)|_{\mathrm{Im}}\right) . \tag{2.10}$$

Numerically, we implement and train the ML model that minimizes Equation 2.10 in PyTorch, a Python package for building deep learning and other machine learning models [128]. This platform is flexible to our unique architecture, which lacks the abstract layers of a neural network and instead propagates information only through acoustically motivated operations, namely the encoding/decoding of the multistatic data and the generation of a B-mode image with DAS beamforming. In all other respects, we mirror the conventional ML pipeline by minimizing our loss function over batches of training data. We perform the numerical optimization using the Adam algorithm, an adaptive version of standard stochastic gradient descent [89], and accommodate the constraint set Σ for physically realizable encoding sequences with a simple projection of our sequence during each update. The update loop of our ML model is depicted in Figure 2.1, for which we emphasize the portion that is deployed in an experimental context.

To effectively update the parameters of any ML model requires knowledge of first order derivatives, which we obtain using the reverse-mode automatic differentiation, or *backpropagation*, capabilities of PyTorch [55]. In the context of ultrasound imaging, standard techniques are often sufficient when the loss function depends primarily on multistatic data [26] or on individual pieces of RF data [122]. This is because platforms like PyTorch have built-in analytic derivatives for operations that are common in deep learning applications, such as matrix multiplication and Fourier transforms. Gradients for all other operations can be constructed at runtime at the cost of additional computation time and memory overhead. However, this process is made difficult when an ML

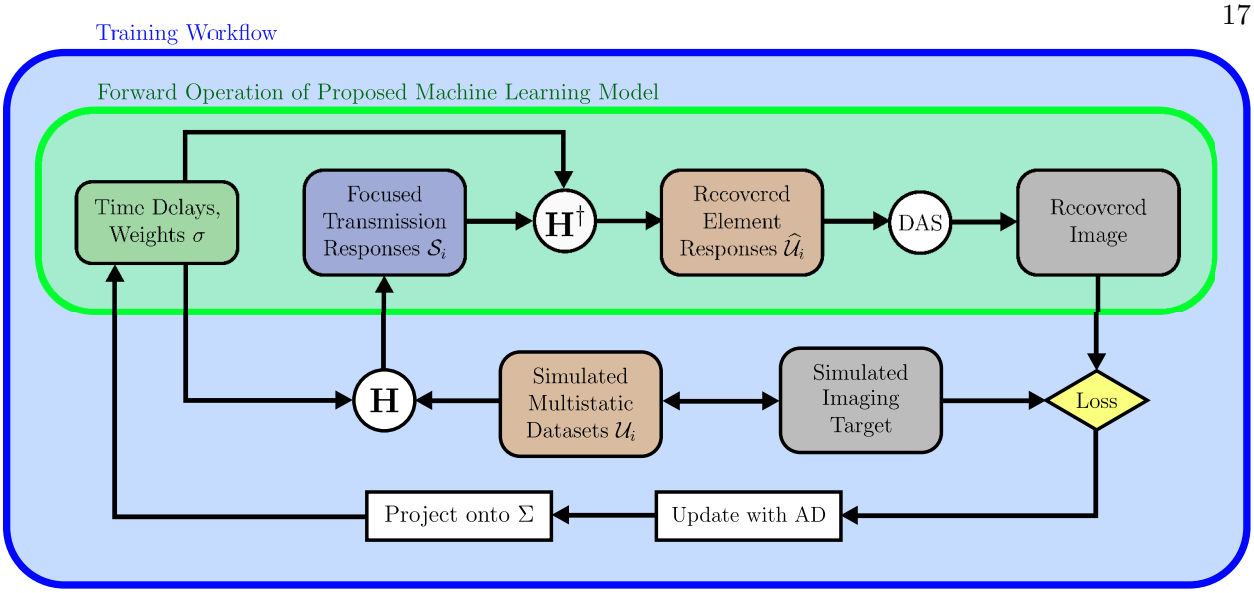


Figure 2.1: Training procedure for proposed ML model for data acquisition and imaging. Simulated multistatic training data is encoded and decoded according to our model parameters, an encoding sequence, and an image is formed using delay-and-sum beamforming. The resulting B-mode image is evaluated by comparison to a target image, or with some other data obtained during the simulation, i.e., target position.

model directly incorporates beamforming, which we consider a critical component to produce high-quality encoding sequences (See Section 2.5.4). Backpropagation through the DAS beamformer rapidly becomes a computational bottleneck when it is implemented directly as a sum of linear interpolations of IQ data into per-pixel focused channel data, as the number of intermediate values that must be stored in memory grows rapidly. This is especially relevant while training the proposed ML model, as we must simultaneously perform this expensive calculation over entire batches of training data.

In our application, we circumvent this issue through a novel PyTorch implementation of the derivative of the DAS beamforming operator $\mathcal{B}$. Because the beamformed image $\mathcal{I}$ is linear as a function of the multistatic data $\mathcal{U}$ [130], the analytic derivative needed for backpropagation is simply its adjoint operator. We provide PyTorch the DAS adjoint for arbitrary data, which eliminates the additional cost and memory overhead.

We do not directly construct and transpose the DAS beamforming matrix, as it is impractically large for reasonably sized multistatic data, even under a sparse representation [130]. Instead,

we apply this adjoint matrix-free, recognizing that $\mathcal{B}$ can be described as a sum of linear interpolations, or

$$\mathcal{B}(\boldsymbol{\mathcal{U}}) = \text{Sum}(\text{Interpolate}(\boldsymbol{\mathcal{U}}))\,.$$

From this, we derive from simple analytic formulas that

$$\mathcal{B}^*(\mathcal{I}) = \text{Interpolate}^*(\text{Sum}^*(\mathcal{I}))\,.$$

These two component adjoints have known, albeit obscure formulas [32], and so we provide pseudocode for both operations for clarity. The code used to perform the optimization, along with the RF data that support the findings of this study, are available at `github.com/jcs15c/optimal_ultrasound_encoding` [168].

Algorithm 1: DAS Beamformer $\mathcal{B}$

Input: $\boldsymbol{\mathcal{U}}$: Multistatic Data
Output: $\mathcal{I}$: Beamformed Image
Initialize focused IQ data at each pixel to zero
for each transmit-receive element pair do
    /* Linearly interpolate per-pixel time delays onto the channel data for the transmit-receive element pair */
    for each pixel in the image do
        Identify the pair of IQ data the pixel focused time delay is between.
        Add a linear combination of these two IQ data to the focused IQ data

2.3.4 Acquisition of Training and Testing Data

While training on true *in vivo* data would best reflect the experimental setting, it is understood that large collections of general purpose, labeled clinical data are rare [100]. Instead, we utilize simulated multistatic data created with Field II [83] to train and evaluate our proposed ML model. The simulated transducer configuration is described in Table 2.1. We have found that multistatic data depicting randomly placed anechoic lesions in an underdeveloped speckle pattern is a particularly effective class of training data, as encoding sequences trained on it generalize well to other classes of data (See Section 2.5.2).

Algorithm 2: DAS Beamformer Adjoint $\mathcal{B}^*$

Input: $\widetilde{\mathcal{I}}$: Data of the same shape as $\mathcal{I}$
Output: $\widetilde{\mathcal{U}}$: Data of the same shape as $\mathcal{U}$
Initialize output to zeros
for each transmit-receive element pair do
    `/* Perform adjoint-interpolation of` $\widetilde{\mathcal{I}}$ `based on per-pixel time delays */`
    for each pixel in the image do
        Identify pair of IQ data the pixel focused time delay is between
        To each index of the output where the closest pair is located, add the corresponding value of the linear combination in Algorithm 1

For training, we generate with Field II a collection of 500 instances of underdeveloped speckle data, of which 20% form a validation set for by-hand tuning of our (reasonably few) hyperparameters. Notably, the proposed model is exposed *only* to this type of data during training. For thorough testing of the optimized encoding sequence, we use additional samples of underdeveloped speckle data, as well as collections of other types of simulated data. These include conventional imaging targets, namely isolated point scatterers and anechoic lesions in fully developed background speckle. These data are also used to evaluate the optimized encoding sequence according to standard ultrasound imaging quality metrics, such as the cystic resolution and gCNR respectively.

To explore performance on more complex RF data, we follow a procedure similar to [75] and generate arrangements of scatterers with amplitudes weighted according a grayscale image. These images are derived from a set of 160 samples from the publicly available validation set of an ImageNet competition [71], where we have extracted and smoothed the middle 256×256 pixels from each and converted them to grayscale. The first type of image-derived data interpolates the pixel values of each grayscale image to nearby scatterers that are distributed throughout the spatial domain, allowing us to consider data with scatterers of arbitrary amplitude. The second type of image-derived completely removes scatterers near pixels of sufficiently low brightness, resulting in anechoic regions in background speckle that are more arbitrary than standard circular cysts.

We also test our optimized encoding sequences on experimental hardware with both a tissue-mimicking phantom and a wire target, as described in Section 2.4.4.

We must similarly consider the imaging target used by the loss function in Equation 2.8 during the supervised learning. A natural choice is an "unencoded" image, where each simulated ground-truth multistatic data set is directly focused with DAS beamforming, thereby removing artifacts induced by encoding. Alternatively, one can compare to a more "idealized" target that represents the ground-truth *contrast*, as this also reduces the influence of any particular speckle pattern during training. Each image of ground-truth contrast is generated alongside the associated simulated multistatic data, where spatial masks are used to create homogeneous anechoic and scattering regions. An alternative approach to describing ground-truth echogenicity would be to directly convolve a high-quality point spread function (PSF) with each scatterer [54], but we find our approach using a spatial mask to be simpler computationally. For image-derived data, the ground-truth contrast is simply the original grayscale image. During training, we use ground-truth contrast targets, and discuss the consequences of this decision in Section 2.5.3.

In Figure 2.2 we show an example of each target type for each category of simulated data, highlighting the particular configuration of data that is exclusively used during training.

2.4 Results

2.4.1 Optimization of Both Time Delays and Apodization Weights

We now demonstrate our capability to effectively train the proposed ML model, thereby producing encoding sequences that meaningfully improve image quality. In the most general case, the available hardware allows us to manipulate both time delays and apodization weights for each array

Parameters	Simulated array
Element count	64
Element pitch	0.3 mm
Element kerf	0.01 mm
Center frequency	3 MHz
Bandwidth	70%

Table 2.1: Simulated Field II Transducer Parameters.

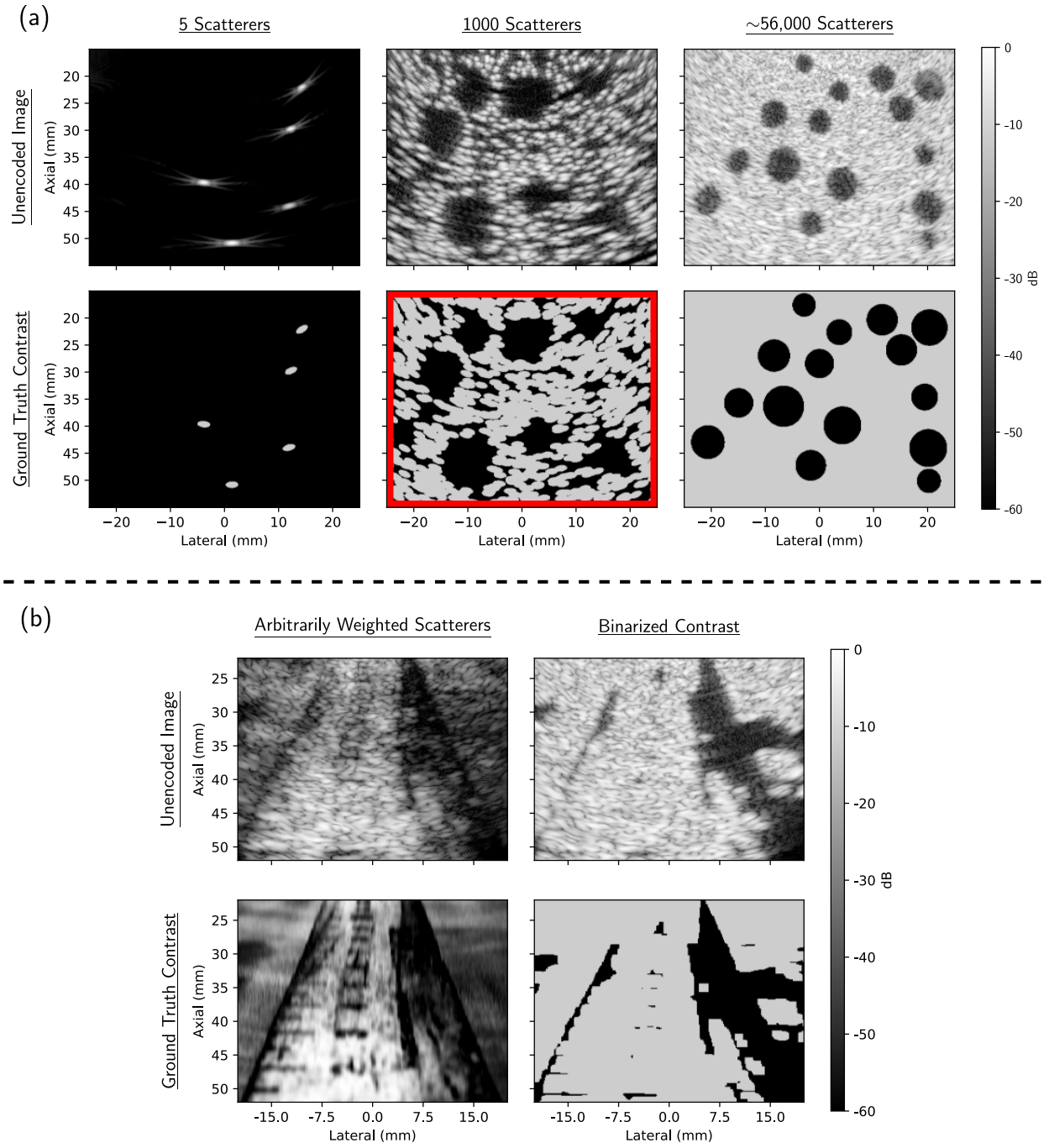


Figure 2.2: Types of simulated data and imaging target type on which the ML model is trained and/or evaluated. (a) Conventional imaging targets generated from the uniform responses of individual scatterers in an anechoic field. (b) Imaging targets for image-derived data, where the amplitude of each scatterer is weighted according to a grayscale image. While all categories are used for validation of the ML model, numerical experiments suggest that training on the class of data emphasized in red (ground-truth contrast, underdeveloped speckle) produces the best optimized encoding sequences.

element. This allows the greatest freedom when designing encoding sequences, and predictably the greatest improvement over conventional sequences. As a prototypical configuration of the training workflow, we train the ML model using 400 simulated instances of the previously described under-developed speckle background with anechoic lesions, processed in batches of size 8. We evaluate the model with the ℓ^2 loss function in Equation 2.8 using ground-truth contrast as the imaging target. The training is performed using the Adam optimizer using a learning rate of 0.1 over 25 epochs. The available Verasonics hardware is limited to a minimum duty cycle of two clock cycles, which effectively nullifies low values for our weights. As a result, we appropriately restrict the parameter space of encoding sequences Σ during training (See Section 2.3.3).

While this is one of many possible configurations of the proposed training workflow (e.g., the choice loss function can vary according to the specific imaging task) we will see that the optimized encoding sequence created in the above configuration improves image quality fairly broadly across a number of different metrics.

We first demonstrate these improvements by evaluating the optimized encoding sequence on several types of simulated ultrasound data in Figure 2.3, for which we uniformly observe significantly improved contrast and reduction of scattering artifacts. For comparison, we use two conventional planewave encodings of varying maximum angle extent, as such sequences characterize a common tradeoff between FOV and resolution at depth [20]. For example, the first generates 15 beams with 1 degree of separation, which results in a higher resolution along a narrower FOV. This is particularly notable when imaging point targets, where some scatterers are essentially undetectable. Steering these 15 planewaves sufficiently far apart to cover the entire imaging domain results in shallower contrast for anechoic regions, as there is more significant scattering throughout the domain.

By contrast, the proposed ML model has generated a sequence that recovers this wider FOV simultaneously with improved contrast. Importantly, we observe through the improvements on the image-derived data that the optimized encoding sequence is performant regardless of the specific arrangement of scatterers in the field. Instead, the point spread function for each scatterer is improved more generally and consistently throughout the FOV.

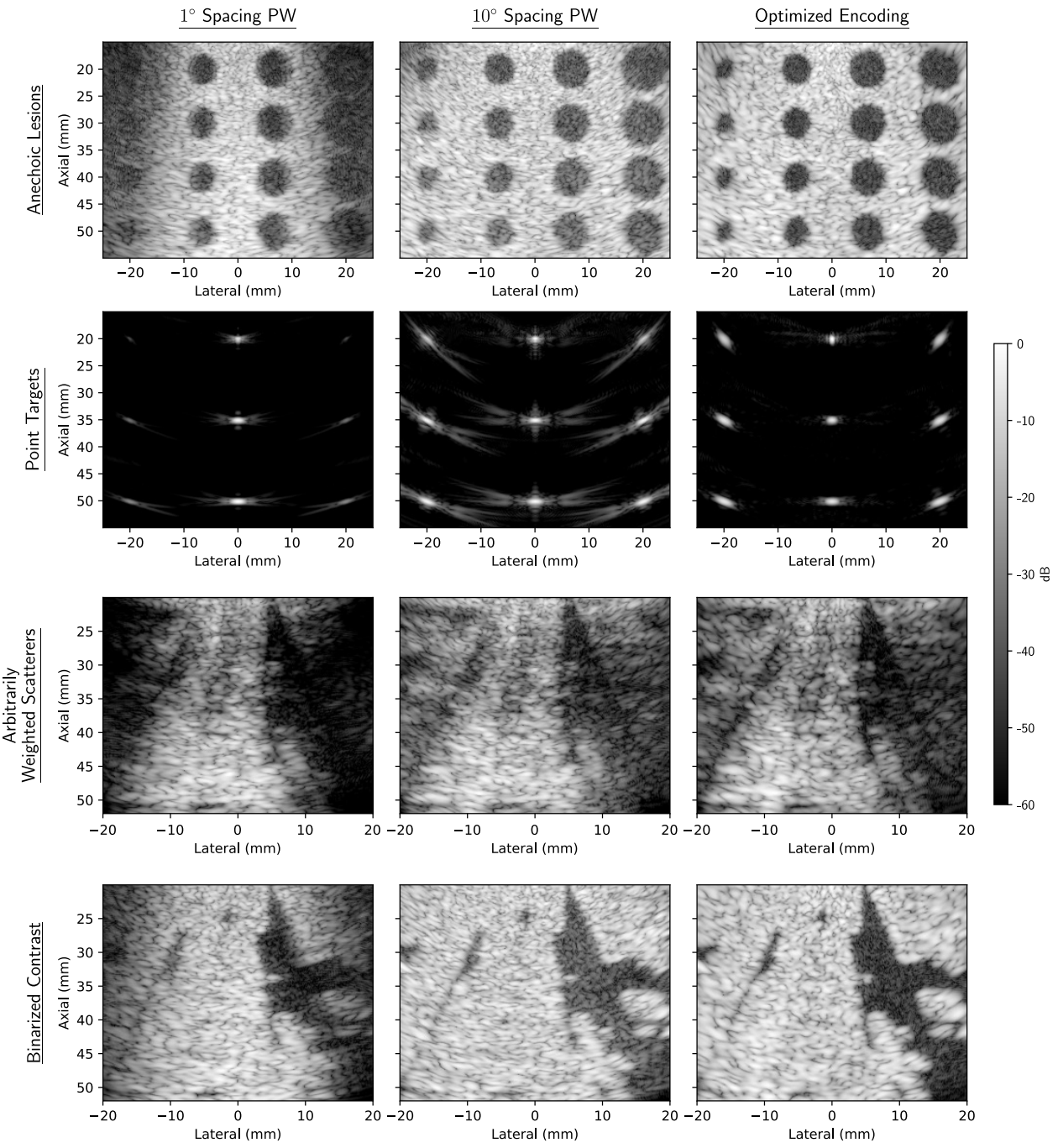


Figure 2.3: Comparison of 15-transmit encoding sequences applied to different imaging targets. We consider imaging using planewaves with 1 degree of separation (left), planewaves with 10 degrees of separation (middle), and our novel optimized sequence (right). In all types of simulated data, we see considerable improvements in contrast and resolution over the conventional transmit encodings.

We show these improvements quantitatively in Table 2.2, where this prototypical optimized encoding sequence leads to improved ℓ^2 error against the ground-truth contrast. This is the same metric that is minimized over the training set of anechoic lesions in underdeveloped speckle (See Equation 2.8), and it is therefore natural that our ML model improves this metric for this type of data. However, we see that these improvements in the ℓ^2 loss are consistent across several *other* types of imaging targets that are visually quite distinct.

Average ℓ^2 Loss against Ground-Truth Contrast Map				
Target Type	1° Spacing Planewaves	Full FOV Span Planewaves	Truncated Hadamard	Optimized Encoding
Isolated Point Targets	12.55	20.03	33.87	**8.12**
Anechoic Lesions in Underdeveloped Speckle	583.9	331.6	366.3	**252.9**
Anechoic Lesions	530.8	250.5	257.3	**205.2**
Image-Derived Contrast	256.6	166.8	174.4	**160.8**
Binarized Image-Derived Contrast	369.9	258.8	274.7	**230.1**

Table 2.2: Average ℓ^2 loss against the ground-truth contrast across 50 pieces of out-of sample simulated data. Although the encoding sequence is trained using only the data set of anechoic lesions in underdeveloped speckle, this metric is improved for each other type of data, emphasized in bold.

We can also demonstrate that improvements in this ℓ^2 loss are strongly associated with improvements in more conventional image quality metrics, namely the gCNR and cystic resolution, thereby justifying the decision to select an ℓ^2 loss for training of the ML model.

The cystic resolution quantifies detectability of an anechoic cyst in background speckle with a given level of contrast [138]. We compute this metric through cystic *contrast*, which is the concentration of energy within a certain distance from a point target in an anechoic field. When cystic contrast is considered as a function of target radius, the cystic resolution is the minimum radius that achieves a desired level of contrast. For our examples we use a contrast of -20 dB, a common threshold for lesion detectability that represents an order of magnitude difference from

the surrounding speckle. By explicitly plotting the cystic contrast for each encoding sequence in Figure 2.4(a), we can see that the optimized encoding sequence produces lesion detectability on par with narrowly concentrated planewaves, and that this conclusion is not sensitive to the exact choice of threshold.

In contrast to these narrow planewaves, however, our optimal encoding sequence maintains this level of resolution throughout the imaging domain. To see this, we use Field II to simulate the responses from single, isolated scatterers distributed throughout the viewing window, and compute the cystic resolution for each when imaged with the optimized sequence. Shown in Figure 2.5, this procedure generates a map of lesion detectability as a function of target position. This strongly suggests that the benefits of our optimized encoding sequence stem from more general acoustic principles, rather than overfitting to the training data, as the ML model is never exposed to these isolated point targets during training.

We additionally measure contrast using the gCNR, which also varies according to the spatial position of the lesion target. To account for this variance, we consider simulated anechoic lesion data for which the position and radius of each lesion is random. We categorize these lesions based on position, and tabulate the average gCNR for each category in Figure 2.5. To consider more complicated lesion shapes, we also measure the gCNR of the binarized, image-derived data of Figure 2.2(b), which we record in Table 2.3. Altogether, we see that our optimized encoding sequence outperforms conventional planewave and Hadamard encodings in terms of both FOV and contrast.

	Average gCNR			
Target Type	1° Spacing Planewaves	Full FOV Span Planewaves	Truncated Hadamard	Optimized Encoding
Binarized Image-Derived Contrast	0.882	0.654	0.888	**0.891**

Table 2.3: Average gCNR across 50 pieces of out-of-sample simulated data, for which the ground-truth contrast is derived from a grayscale image. Observe that our optimized encoding results in the highest gCNR, emphasized in bold.

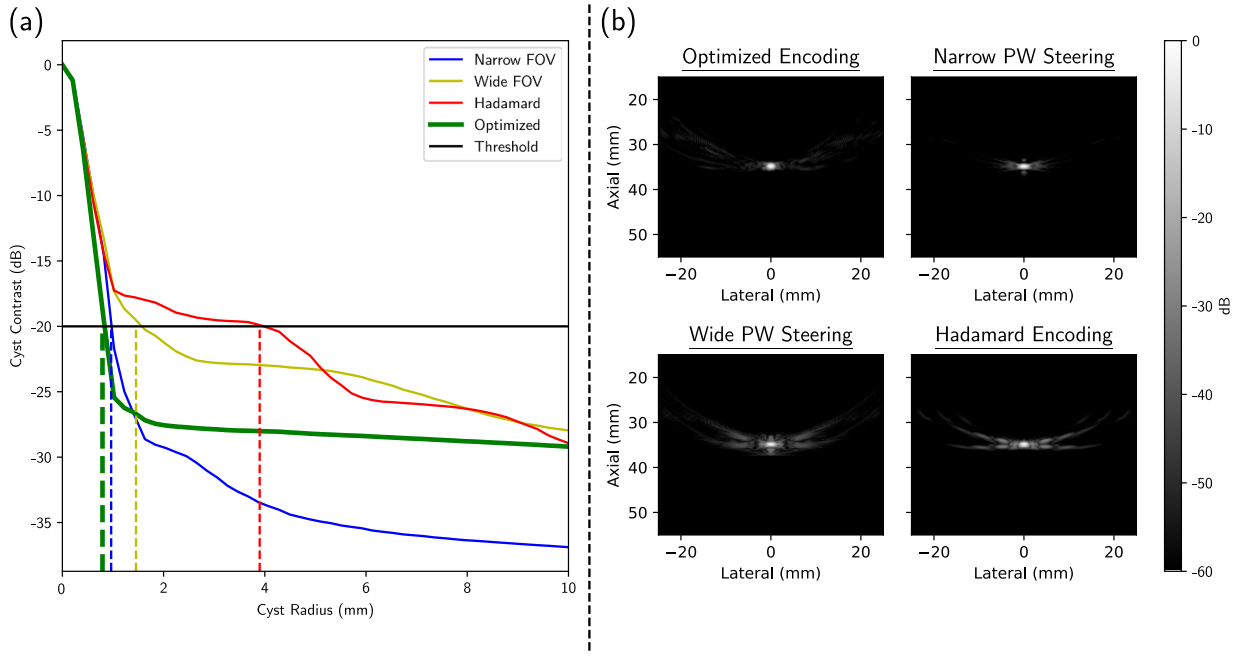


Figure 2.4: Comparison of cystic contrast for different encoding sequences. (a) We plot the contrast of an anechoic cyst as a function of cyst size, with a vertical line indicating the value of the cystic resolution. (b) The centered point targets for which the cystic resolution is measured. From this, we can see that the optimized encoding results in cystic resolution comparable to that of narrow span planewaves.

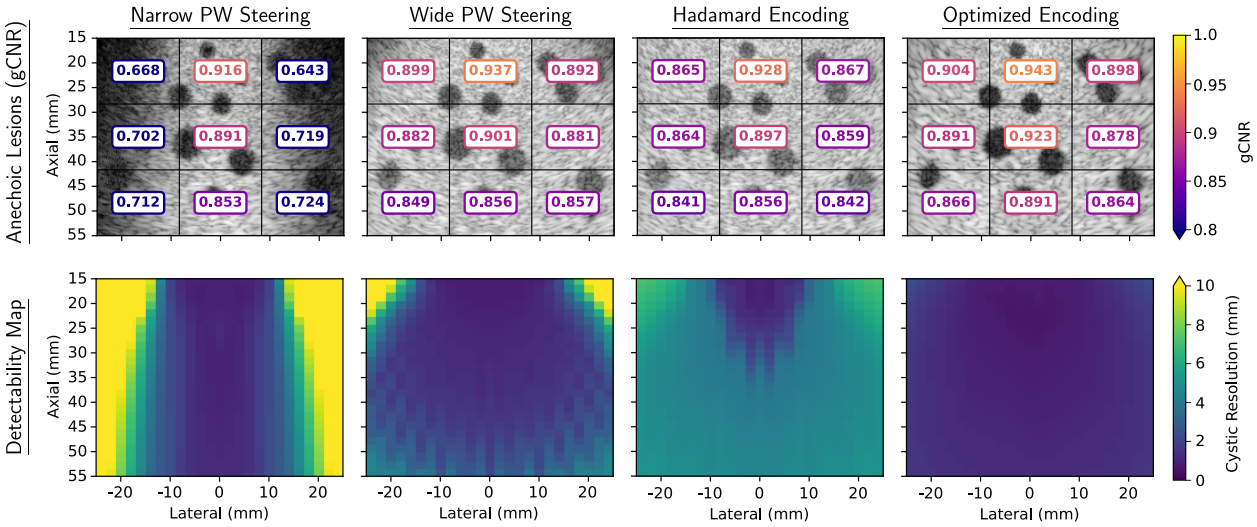


Figure 2.5: Comparison of spatially aggregated gCNR and cystic resolution for different encoding sequences. (top) We plot the average gCNR for anechoic lesions in each region of the viewing range, averaged over randomly located targets over 50 instances of simulated data. Images are displayed for one instance of randomly located targets. (bottom) We plot the cystic resolution throughout the domain. Observe that only our optimized encoding sequence is able to maintain the same quality measured by gCNR and cystic resolution throughout the imaging domain.

We also compare our optimized sequence to standard techniques in apodization, which similarly seek to improve the PSF by reducing side lobes [85]. To do this, we consider planewaves with 10° spacing, which offers good resolution at depth while imaging the full FOV, and apply a Tukey window to the receive channel. As expected, this improves cyst detectability in the image considerably. However, it does so at the cost of resolution, as shown in Figure 2.6. Specifically, there are still clear artifacts around point scatterers, and the striation in the cystic resolution map for the unapodized case is still present after apodization. These effects are not present in our optimized encoding sequence.

2.4.2 Restricted Optimization to Either Time Delays or Apodization Weights

We can also apply the proposed ML model and training framework to more restrictive hardware specifications. In doing so, we demonstrate that there is still considerable improvement present, although such sequences do not perform as well as a fully arbitrary system, as expected. To this end, we train two separate configurations of the ML model. In the first, the apodization weights are fixed to some uniform value, and the time delays are the *only* trainable parameters of the ML model. In the second, only the apodization weights can be changed, and each time delay is fixed to zero in the ML model. As in Section 2.4.1, we train the model on anechoic lesions in underdeveloped speckle and test the resulting sequence on data far outside the training set. We compare the two resulting optimized sequences to the appropriate conventional alternative: wide spanning planewaves for delay-only optimization, and a truncated Hadamard encoding for weight-only optimization.

The results of this comparison are compiled in Table 2.4. We see that in each case the optimized sequence performs better according to the ℓ^2 loss metric than its conventional alternative, which we have now shown correlates with high quality as measured by gCNR and cystic resolution. Furthermore, we observe that while there is still considerable improvement when only delays are optimized through the training of our ML model, the magnitude of improvements when only optimized weights are generated is comparable to that of a fully optimized encoding sequence.

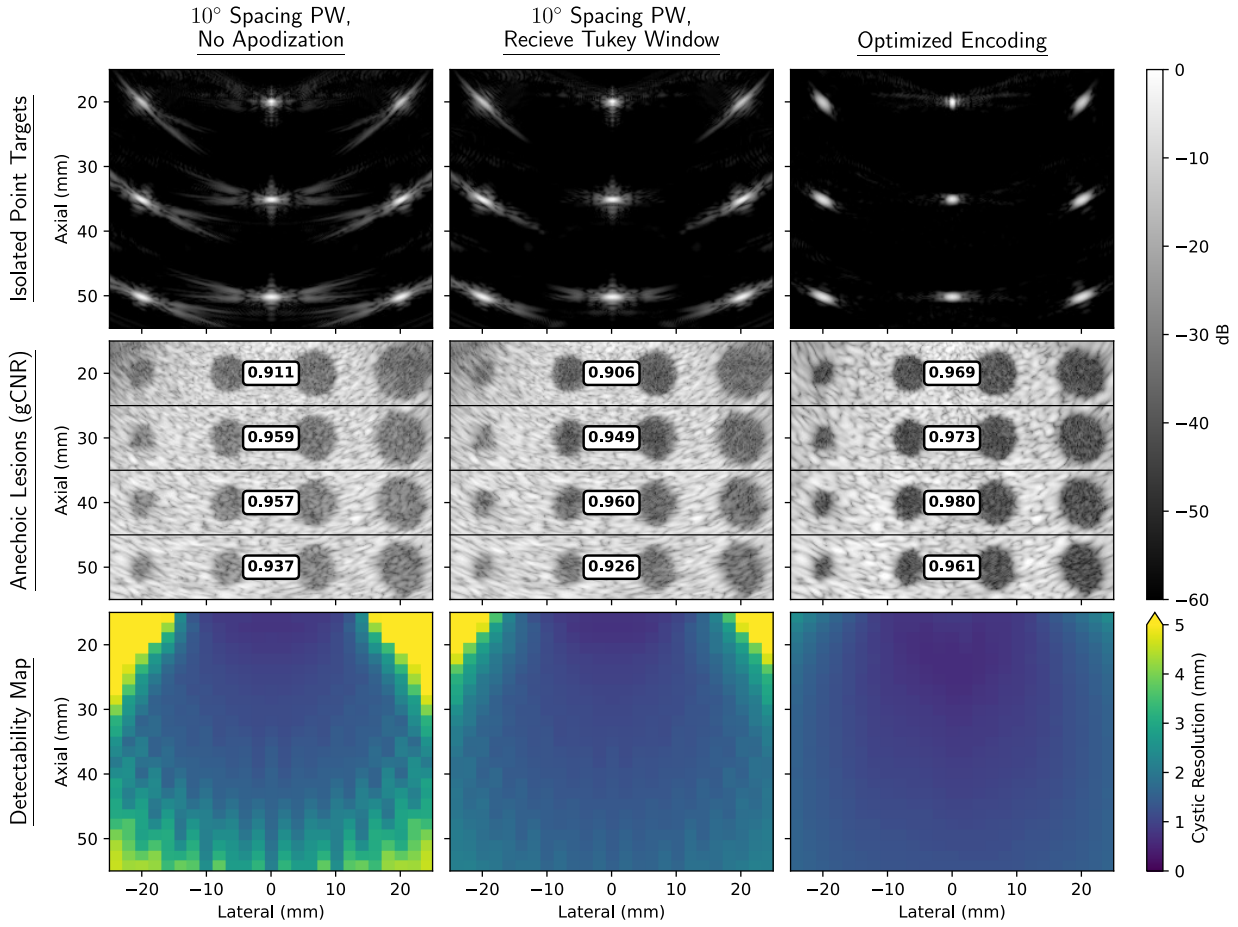


Figure 2.6: Comparison to application of classical apodization. We do this visually (top), through the average gCNR across each row of lesions (middle) and through the cystic resolution (bottom). We observe that although the Tukey window does improve the PSF, there is greater improvement from our optimized sequence.

This demonstrates that in this imaging scenario, optimization of the weights has a more significant influence on the quality of the resulting sequence than delay optimization alone.

Average ℓ^2 Loss against Ground-Truth Contrast Map					
	Delay Optimization		**Weight Optimization**		**Both**
Target Type	Optimized	Full FOV Planewaves	Optimized	Truncated Hadamard	Optimized
Isolated Point Targets	**16.93**	20.03	**14.53**	33.87	**8.129**
Underdeveloped Speckle	**310.2**	331.6	**303.3**	366.3	**252.9**
Anechoic Lesions	**231.2**	250.5	**217.1**	257.3	**205.2**
Image-Derived Contrast	**165.7**	166.8	**164.2**	174.4	**160.8**
Binarized Image-Derived Contrast	**243.3**	258.8	**241.2**	274.7	**230.1**

Table 2.4: Comparison to ML models whose trainable parameters are restricted to only a single component of the encoding sequence. In both cases, we see that the per-component optimized encoding sequence outperforms the conventional alternative on each type of imaging target.

2.4.3 Optimization in the Presence of Noise

These simulated results demonstrate the theoretical utility of applying an ML optimization strategy for data acquisition and imaging within the REFoCUS framework. However, we can also verify that our optimized sequences produce meaningful improvements in several practical scenarios. In the first, we consider a separate configuration of the ML model that is trained to emulate an imaging system under the influence of electronic noise. We recreate this during training by the addition of random noise to the encoded channel data prior to imaging. In the current example, this additive noise is sampled from a random normal distribution, filtered to have a bandwidth equal to that of the transducer, and then scaled to produce an average of 20 dB channel signal-to-noise ratio. It is in this context that restricting the set of apodization weights to an ℓ^∞ ball is most critical, as otherwise all noise would be suppressed simply by arbitrarily increasing the apodization

weights.

By accounting for this noise during training, the ML model generates a new encoding sequence that specifically suppresses this noise. We see this in comparison to the optimized encoding sequence evaluated throughout Section 2.4.1. In Figure 2.7(a), we consider these two sequences, trained with and without the presence of noise, along with a conventional planewave configuration for comparison. We then apply each sequence to data to which noise is, or is not added. In each of the four cases, we see considerable improvement over the conventional planewave approach. However, we can also see that these improvements are most prominent when the ML model is trained with the same type of noise present during evaluation, the case emphasized in Figure 2.7(a) in red. In Figure 2.7(b), we consider a more complete evaluation of the case where noisy data is imaged using a sequence trained with the same noise model. There is the same degree of improvement as the noiseless case of Figure 2.3, featuring dramatic improvements in the gCNR and cystic resolution.

2.4.4 Experimental Verification

Finally, we verify these results by directly implementing the optimized encoding sequence discussed in Section 2.4.1 in physical hardware. In the following experiments, we use the Verasonics Vantage 256 research scanner (Verasonics, Inc., Kirkland, WA) with the P4-2v phased array transducer (3 MHz transmit frequency, 64 elements, 0.3 mm pitch, 10V transmit voltage). Received channel data were stored for processing offline, where they were decoded and used to produce an image in the same manner as in simulation. For comparison, we use sequences of 15 transmits that generate planewaves of varying span (15, 60, and 150 degrees). To reproduce the effects of arbitrarily scaled weights (rather than restricting to ± 1 through pulse inversion) we use the Verasonics apodization function to scale the duty cycle of the transmit excitation so that it matches the output amplitude.

The first experiment utilizes an ATS 539 multi-purpose phantom (CIRS, Inc., Norfolk, VA) to confirm the ability of optimized encoding sequences to improve imaging on tissue-like material. Within the phantom we image anechoic lesions of varying position and radius in background speckle,

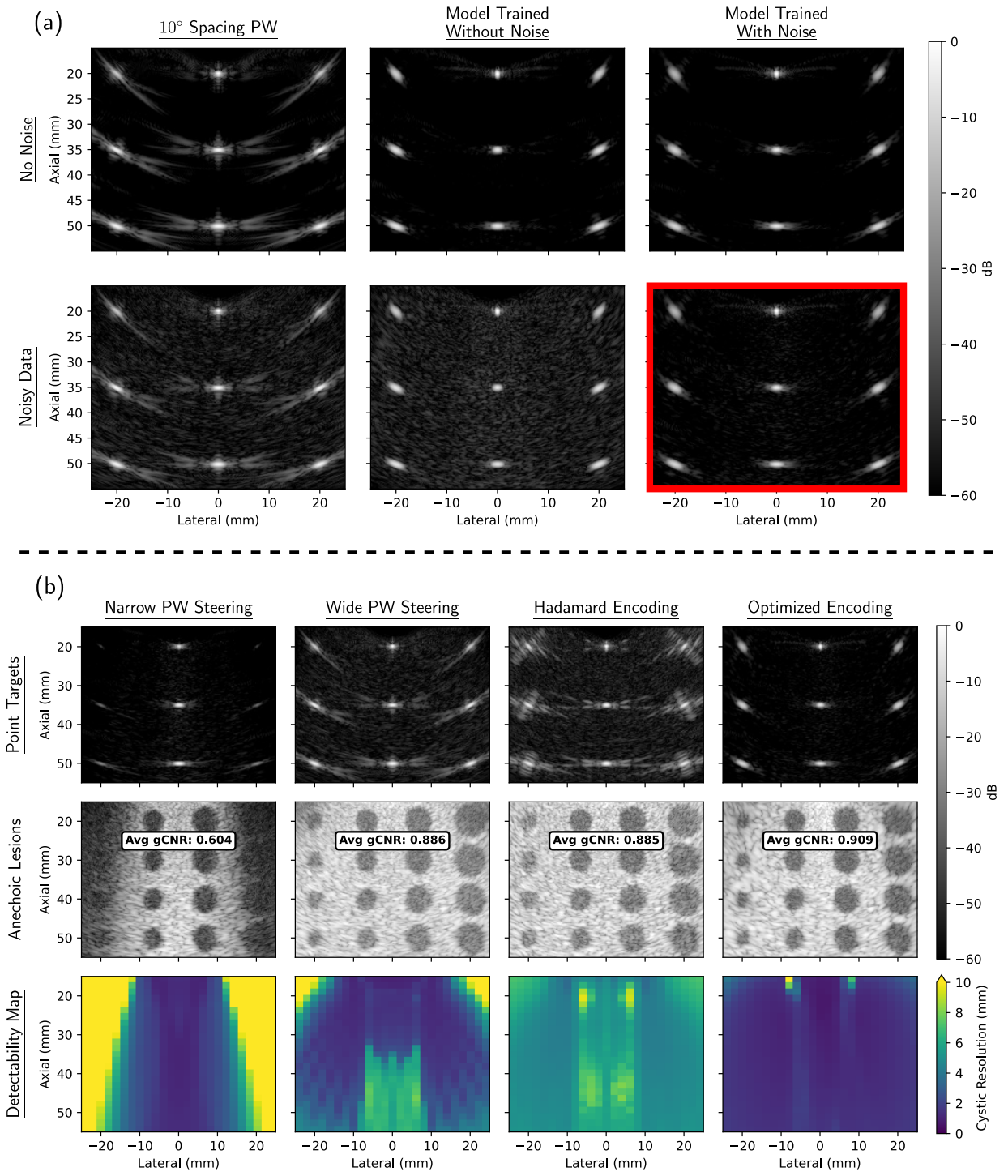


Figure 2.7: Effects of noise on training/evaluation of an ML model. (a) We compare encoding sequences generated by different ML models trained with and without the presence of electronic noise. (b) We make additional comparisons to the case emphasized in red in (a), comparing conventional transmit sequences on noisy RF data with a sequence optimized in the presence of additive noise. We see a greater qualitative suppression of noise (top), as well as improvements in gCNR (middle) and the point spread function as measured by the cystic resolution (bottom).

and compute the gCNR of each with respect to surrounding background speckle. We show these gCNR values below each lesion in Figure 2.8(a). For a fixed lateral position, we acquire data using each encoding sequence at each of seven elevation positions of the phantom. In this way, we image anechoic lesions in the same positions with different realizations of background speckle, and compute the average gCNR across all realizations.

We can see that these results match closely with our numerical simulations, in that our optimized sequence is able to produce the depth of focus and resolution of a very narrow span of planewaves, while still maintaining a full FOV throughout the range. Importantly, these improvements persist beyond the fixed viewing window of the training data, which extends only to a depth of 55 mm. This further emphasizes that our encoding sequence has not simply specialized to the specific characteristics of the training data.

We can also recreate our simulated point target images, which we can in turn use to measure the cystic resolution throughout the imaging domain (albeit on a much coarser grid). To accomplish this, we image a custom, single target wire phantom (0.03 mm tungsten wire) in a water tank. Using the AIMS III Hydrophone Scanning System (Onda Corporation, Sunnyvale, CA), we mechanically translate the suspended P4-2v transducer around the fixed wire so that the target can be imaged from a 6×5 grid of target positions (∼10 mm lateral spacing, ∼8 mm axial spacing).

Although we only ever image one point target at a time with this experimental setup, in Figure 2.8(b), we present a *montage* of these images distributed appropriately in space. To ensure a fair visual comparison of each point target, we apply a *uniform* level of gain to each image in the domain. This allows us to very clearly see the effect of our optimized sequence on the FOV of the imaging system, where points in the upper corners of the viewing window are either undetectable or suffer from very poor resolution. In contrast, the optimized encoding sequence provides a uniform degree of resolution at all points in the domain.

Furthermore, we can see that the off-axis scattering artifacts present are greatly reduced when imaged with the optimized sequence. As before, we quantify this effect using the cystic resolution, which we can compute for each point target in Figure 2.8(b). Although our ability

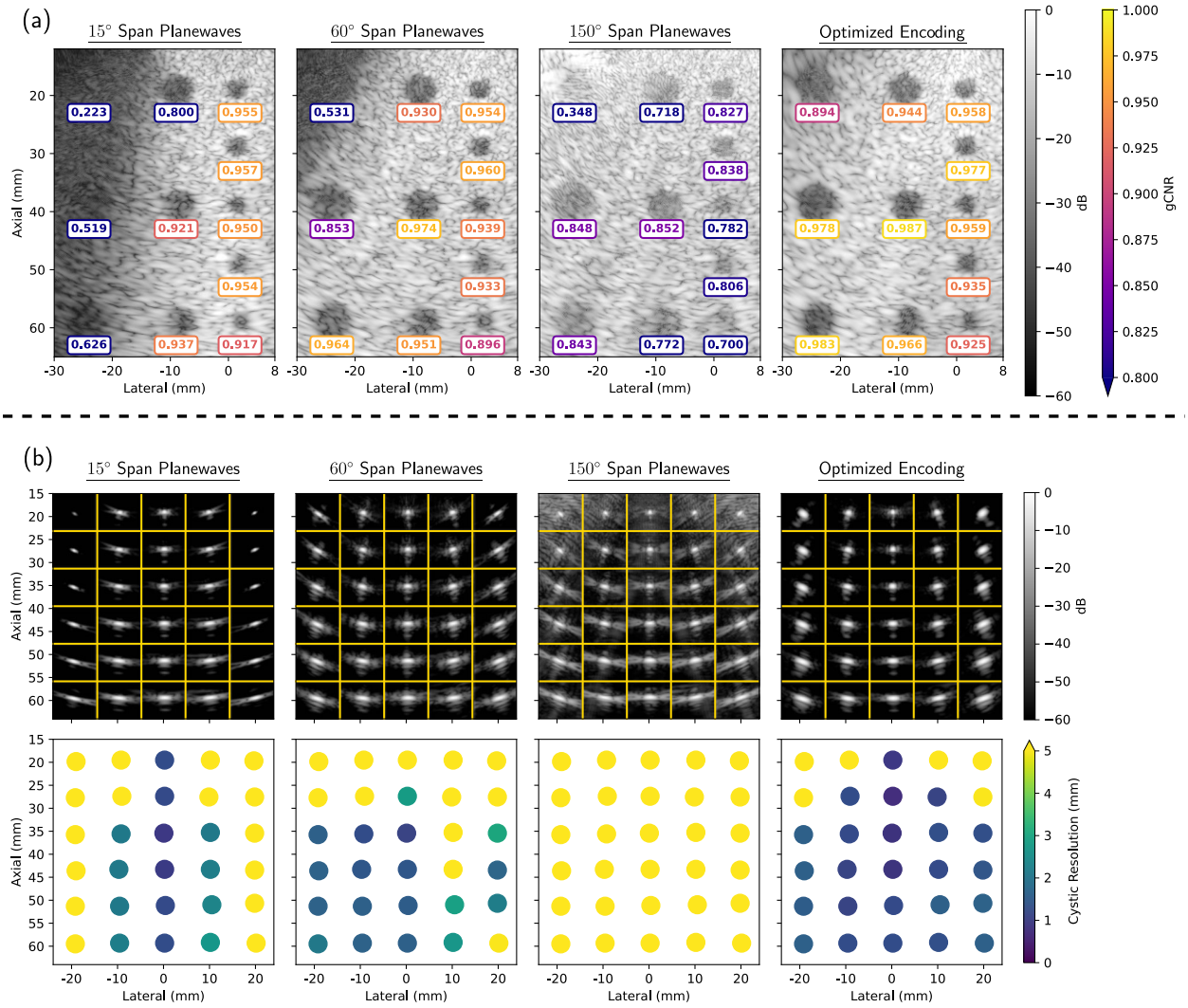


Figure 2.8: Experimental validation of simulated results. (a) We image the ATS 539 multi-purpose phantom using different encoding sequences. Although the ML model is optimized for a symmetric FOV, the fixed lateral position of the transducer was intentionally placed off-center to demonstrate improved FOV, while fully capturing each target in the asymmetric phantom. The average gCNR over seven realizations of the speckle pattern is shown beneath each anechoic lesion. (b) We create a montage of 30 individual point target images. Each is obtained by translating the P4-2v transducer around a fixed wire target in a water tank. To provide an appropriate comparison between different targets within the montage, each image is displayed with the same amount of gain (top). We use these images to compute the cystic resolution throughout the domain (bottom).

to perfectly capture this metric is slightly hampered by irregularities in the experimental setup, we still observe near uniform improvement over each conventional sequence, analogous to results observed in simulation.

2.5 Discussion

Our machine learning framework represents a novel method of generating transmit sequences that ultimately result in higher quality images than current standards. However, as with most machine learning applications, maximizing the efficacy of the optimized sequence requires careful consideration of the ML model configuration. For example, one must select a number of standard ML hyperparameters, such as descent algorithm, batch size, learning rate, etc. Furthermore, this particular usage of ML in ultrasound imaging brings about additional, unique considerations. In this section, we discuss our own exploration of these considerations, and make observations that we believe will be relevant in any future ultrasound ML application that directly incorporates a beamformer into the architecture.

2.5.1 Significance of the Initial Condition

Because the minimization problem presented in Equation 2.10 is not convex, there is no expectation that any optimization procedure can find the global minimum, should one even exist. Instead, the particular local minimum approached by our optimization procedure is highly dependent on the initial condition. Because locally optimal encoding sequences vary in quality, it is important to select an initial condition that is at or near a good local minimum. The interpretability of the parameters of our ML model offers a distinct advantage over conventional deep learning applications in accomplishing this, as we can intelligently explore the effects of different initial conditions.

For example, an instance of our ML model initialized with a planewave encoding will converge to transmissions that are steered in the same way, albeit with less well-defined wavefronts. On the other hand, initializing with a truncated Hadamard code causes the parameters to converge to a

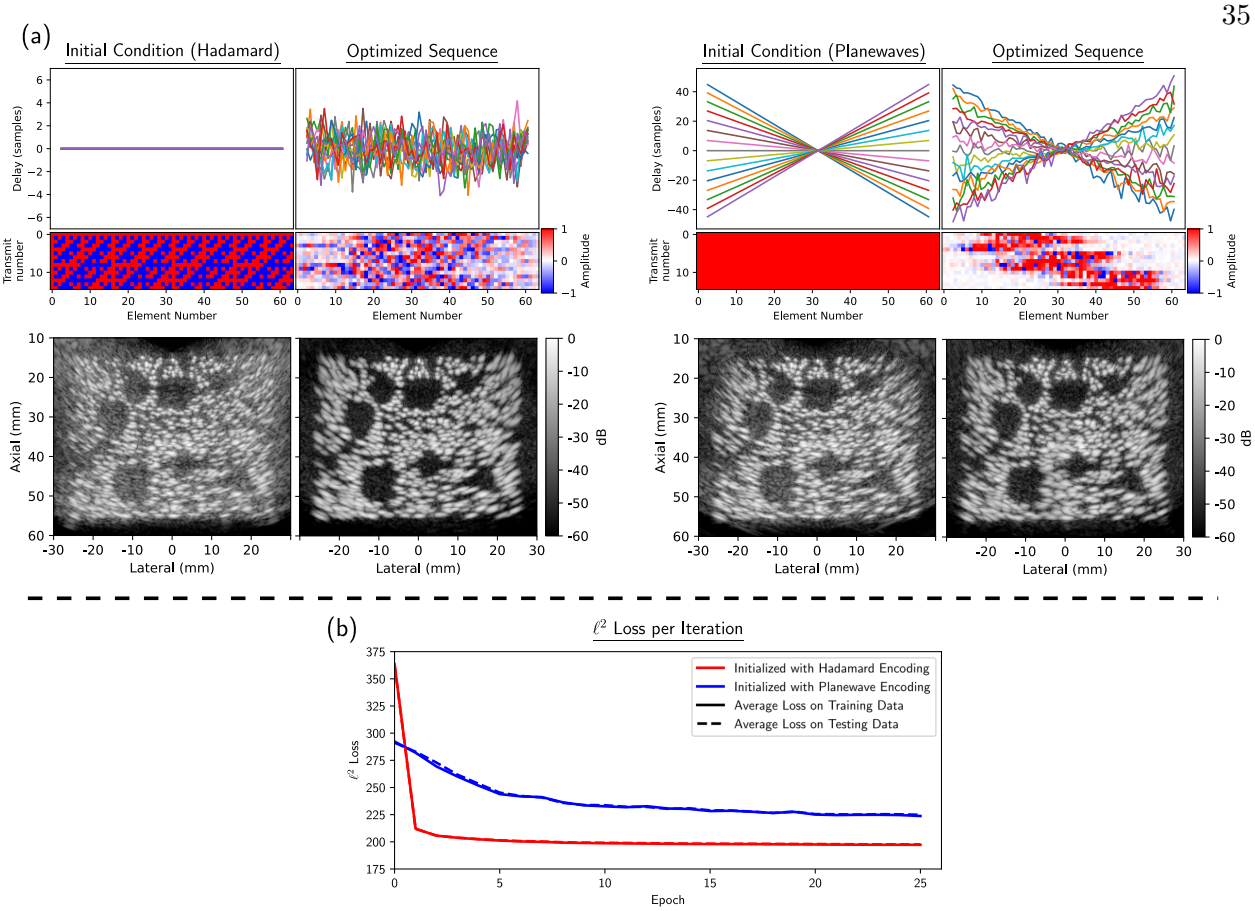


Figure 2.9: Comparison between two different encoding sequences used as initial conditions. (a) For each, we plot the transmit sequence before and after optimization, along with one piece of testing data for each. In the delay plots, each transmission is represented by a single line. (b) For each initial condition, we plot the per-epoch ℓ^2 loss function averaged over the training and testing data sets.

sequence with no visible steering. We can see these qualitative behaviors in Figure 2.9(a). Despite this, we have found experimentally that a truncated Hadamard sequence is the superior initial condition, despite being a generally lower quality transmission sequence. This can be seen through the ℓ^2 loss plotted over the course of training epochs for each initial condition in Figure 2.9(b). This can be explained by the conventional Hadamard sequence allowing the ML model to begin with near-optimal FOV, as well as both positive and negative apodization weights. Except for cases in which delays or weights must be fixed as in Section 2.4.2, each optimized encoding sequence used in this chapter was generated by an ML model with this configuration as the initial condition.

In our exploration of this topic, we have found that recovering a known, conventional sequence

is exceptionally difficult except in the simplest toy problems. For example, although planewave transmissions are generally performant, the ML model cannot identifiably recover the same geometric properties that make them desirable. On the other hand, the optimal characteristics of the resulting sequence are quite opaque: while optimized sequences appear to be a simple perturbation of the initial condition, performing this perturbation randomly fails to achieve the same improvements. This suggests that it would not be possible to discover these sequences *without* the use of the proposed ML model.

2.5.2 Significance of the Training Data

In Figure 2.9(b), we also observe that for either initialization, our ML model performs nearly identically between our training and testing data. This is a highly desired property in this context, as ultrasound imaging can be theoretically formulated as a convolution of a spatially varying point spread function across each scatterer in the domain, the results of which are then summed to form an image. Therefore, an effective manner of optimizing image quality for arbitrary data would be to optimize this PSF itself, concentrating it at zero offset [54].

Although our loss function does not reference features of the PSF directly, our choice to use underdeveloped speckle as training data implicitly encourages the same type of improvement at a lesser computational burden. This is in part because the scatterers within each sample are spread out enough that their off-axis scattering artifacts do not necessarily overlap one another, as would be the case with standard background speckle. This means the ℓ^2 loss function can only be meaningfully decreased by acoustically concentrating the energy of each scatterer inwards, thereby improving the PSF. At the same time, scatterers across *all* samples in the training data are densely distributed throughout the imaging domain. This means that learning improvements that span the entire FOV can be obtained with relatively few samples of type of training data.

2.5.3 Significance of the Imaging Target

Although we exclusively use a loss function that compares produced B-mode images with some ground-truth imaging target, there is still much flexibility within that choice. As stated in Section 2.3.4, a natural choice for the imaging target is an image derived from ground-truth multistatic data. While this produces target images without any encoding artifacts, we have found that an ML model trained with these targets produces poor image quality relative to alternatives. This is because DAS beamforming with even unencoded multistatic data results in a PSF with undesirable features such as the visible side lobe artifacts present throughout the domain. When the ML model is trained on such target images, the transmit sequence is implicitly encouraged to accurately reconstruct these artifacts. Instead, we see that training against the exact ground-truth contrast produces images with a higher quality PSF, as seen in Figure 2.10.

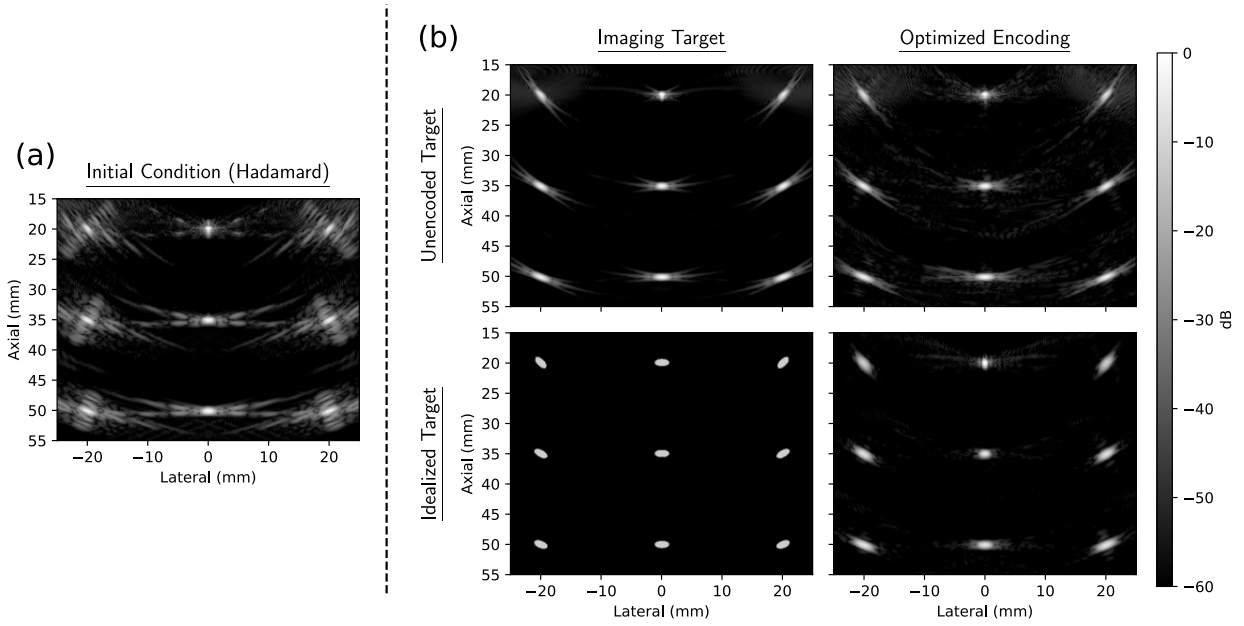


Figure 2.10: Influence of imaging target type on encoding sequence quality. (a) We initialize two ML models identically. (b) When the imaging target of the ℓ^2 loss is unencoded data (top), the resulting sequence has the same artifacts present in the beamformed image. By using an idealized imaging target, i.e., one that represents the ground-truth contrast, the resulting sequence can suppress these artifacts (bottom).

Additionally, we see the most uniform improvement in image quality when we "pad" images of underdeveloped speckle so that the final viewing window is itself located in an anechoic field. This

ensures that the encoding sequence is not unnecessarily dependent on the specific domain geometry. The consequences of this are best seen through the plots of cystic resolution in Figure 2.11, where the first encoding sequence is trained on images whose extent is exactly that of the final viewing window. In contrast, the second encoding sequence is trained on images located in a slightly wider anechoic void. As we can see, adding this empty space causes the PSF to more smoothly vary throughout the domain, and failing to do so results in additional artifacts for point targets near the center of the region. In effect, these streaks depict positions for which scattering artifacts extend beyond the imaging domain, and the introduced error is not captured by the loss function.

2.5.4 Significance of the Loss Function

By using our custom implementation of a differentiable beamformer, we are able to train our ML model according to improvements in an *imaging* metric, which is a novel departure from existing work that only considers the recovery of the multistatic data set. Importantly, we can demonstrate the existence of transmit sequences that produce images with higher resolution and contrast, even though the encoding itself produces a *worse* reconstruction of $\boldsymbol{\mathcal{U}}$ from $\boldsymbol{\mathcal{S}}$.

This is primarily because a perfect reconstruction of $\boldsymbol{\mathcal{U}}$ will only produce the original unencoded image, and this type of image lacks many desirable properties, as discussed in Section 2.5.3. To create a concrete example of this phenomenon, we compare to an ML model within our framework that is instead configured to directly optimize the recovery of the multistatic data set by minimizing

$$\mathcal{L}_{\text{STA}}(\widehat{\boldsymbol{\mathcal{U}}};\boldsymbol{\mathcal{U}}) := \frac{1}{N_T N_R N_\omega}\|\boldsymbol{\mathcal{U}} - \widehat{\boldsymbol{\mathcal{U}}}\|_2^2 \,. \tag{2.11}$$

Consider Figure 2.12. We see that this new configuration successfully generates an encoding sequence that produces a lower STA loss than the previously considered encoding sequence optimized for image recovery. Yet despite this numerical improvement, there is a clear superiority in image quality when the optimization process considers image formation, leading to improvements

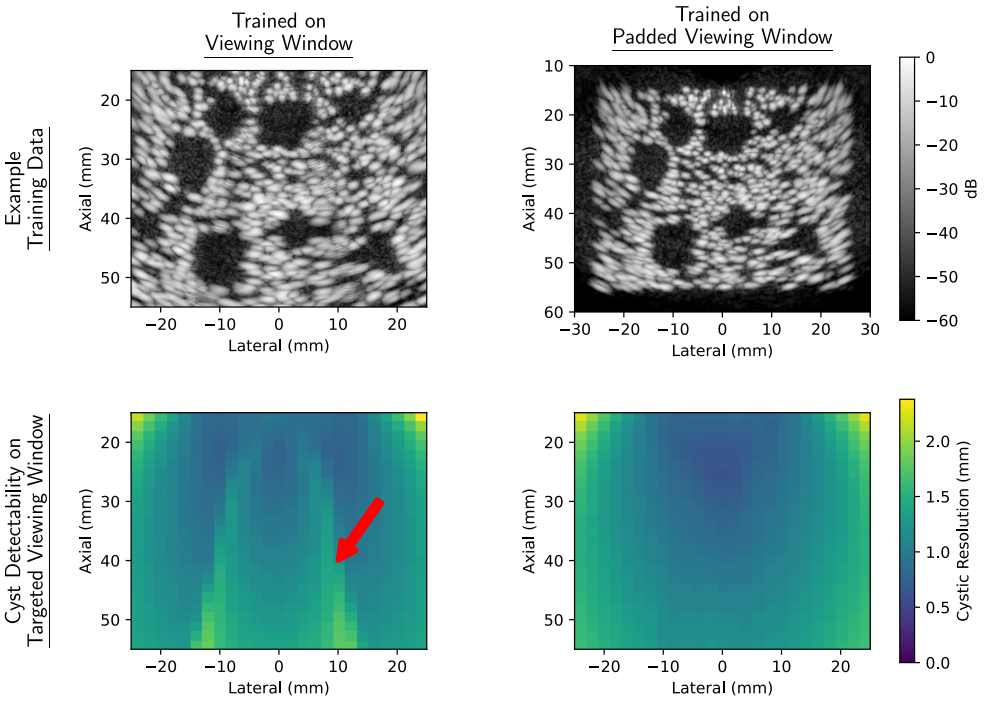


Figure 2.11: Influence of imaging target on image quality. We carefully selected the target to ensure there is not an unnecessary dependence on the specifics of the imaging domain. For example, by expanding the window during training (right), we are able to deal with streaking artifacts (indicated with an arrow) that arise from unsuppressed scattering just beyond the imaging domain.

both visually and in terms of the gCNR averaged over each lesion.

We take this as a counterexample to the common convention that a better conditioned encoding necessarily corresponds to higher quality images. To account for the effects of Tikhonov regularization, we measure the condition number of the collection of encoding matrices $\boldsymbol{\mathcal{H}}$ as $\kappa = \|\boldsymbol{\mathcal{H}}_0\| \cdot \|\boldsymbol{\mathcal{H}}_0^{\dagger}\|$, doing so because the encoding matrix for the lowest frequency $\boldsymbol{\mathcal{H}}_0$ has the poorest conditioning among the collection $\boldsymbol{\mathcal{H}}$. We see that both encoding sequences bring about reasonably well-conditioned matrices following Tikhonov regularization, but there are still clear qualitative differences between the two images.

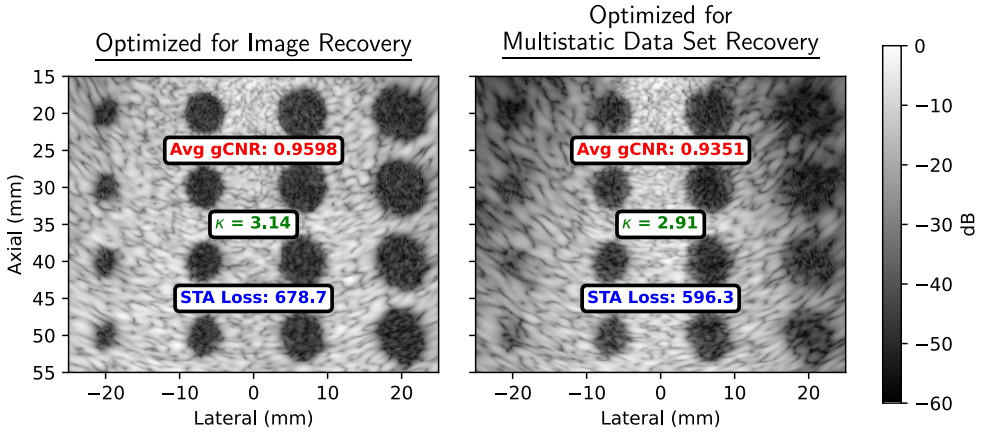


Figure 2.12: Influence of loss function on image quality. We compare our optimized sequence (left) to one that is trained to optimize reconstruction of the multistatic data set (right). In spite of a worse multistatic reconstruction measured by the STA loss and comparable condition number κ for the encoding matrices, our strategy for optimization produces visibly improved contrast and resolution throughout the image.

2.6 Conclusions

In summary, we have shown that the principles of machine learning can be used to generate encoding sequences that improve the quality of ultrasound imaging within the context of the REFoCUS model. An important theoretical consequence of our ML procedure is its ability to discover currently unknown transmit sequences with desirable properties. For example, the various ML models trained for demonstration throughout this chapter have each found transmit sequences that are "high-quality" in a sense that is acoustically meaningful, yet is not directly encouraged by

the training procedure. That is to say, although the sequence is selected so that it minimizes one particular ℓ^2 imaging metric, it ultimately improves quality along a number of other metrics across disparate classes of data. At the same time, the exact acoustic properties that these sequences share remains obscure. This means that our ML model has stumbled upon a currently unexplored regime of transmit sequences, and there remain unanswered questions as to the unifying features of such sequences.

By using an ML model whose parameters are exactly those of an encoding sequence, this approach offers a high degree of flexibility to different imaging scenarios, making it an important foundation for future endeavors. For example, the high degree of generalizability shown in Figure 2.3 indicates that, in sharp contrast to other deep-learning image formation and analysis tasks, it may be possible to train with a limited amount of *in vivo* data without a severe degradation in image quality. The theoretical flexibility of the REFoCUS framework is a key component of this type of investigation, as it allows for uniform treatment of highly variable kinds of transmit sequences. Beyond the proposed framework, we wish to further explore the capabilities of machine learning within REFoCUS, motivated by our ability to incorporate beamforming as a layer of an ML architecture, potentially with its own set of trainable parameters. As has been the case in this chapter, such technology will ultimately allow us to continue developing techniques that directly improve image quality for applications of interest.

Chapter 3

Optimal Experiment Design for Quantum Minimax Fidelity Estimation

3.1 Abstract

Ensuring the proper operation of a modern quantum device requires spending additional resources to verify device output. We consider in this chapter the more general problem of quantum verification through the lens of fidelity estimation, in which measurements of the quantum state inform how "close" a constructed state is to an intended target. This is in contrast to tomography schemes that compute such statistics directly from a fully reconstructed state, as these often require a greater number of measurements to in order to be accurate. To be experimentally viable, a central goal of any method of fidelity estimation is to create an accurate estimate from as few observations, and types of observations, as possible. We present a technique that designs an experimental measurement protocol of a known target state, finding one that minimizes the width of a nearly optimal minimax confidence interval around the true value of the fidelity. Importantly, the nature of the underlying fidelity estimation scheme means that this design procedure is robust to the availability of measurements, and can be designed prior to the collection of any observations.

In collaboration with Akshay Seshadri and Stephen Becker

3.2 Introduction

The verification and validation of existing quantum technologies is a critical component of the current Noisy Intermediate Scale Quantum era. Even when there is advance knowledge of properties of an experimentally acquired quantum state, there are a number of reasons that the true state might deviate from its intended target, and a number of approaches that can quantify this deviation. In this chapter, we are interested in the case where the state is created with a pure target state in mind, but the construction process is subject to noise and other experimentally induced errors [25]. We consider methods which quantify the difference between these states through their *fidelity*, with an emphasis on methods which are indifferent to the source of these errors, as well as to the type of device hardware more generally.

In the tomographic approach to this problem, one uses a classical post-measurement analysis of observables to fully reconstruct the state, such as Maximum Likelihood Estimation (MLE) [81, 49, 77, 9, 1], least-squares [62, 56], or active learning [96]. With this reconstruction of the experimental state, many observables of interest can be directly computed, including the fidelity. However, this often comes at a prohibitively expensive experimental cost, requiring a large number of measurements needed to obtain a reasonably accurate reconstruction, particularly when the fidelity is the only value of interest [18]. Furthermore, schemes to improve the accuracy in the low-measurement scheme necessarily rely on some assumption of the target state, e.g., low-rank, as in [56], or matrix product states (MPS) and other tensor structures, cf. [36, 13, 14, 97, 94, 99, 137]. However, the assumptions are unlikely to exactly hold in reality, so they introduce a bias into the resulting estimate.

Alternatively, there exist more direct "verification" strategies which directly determine if the experimentally acquired state is (up to some tolerance or probability) the same as the target state, ideally using fewer measurements to do so. These methods are often tailored towards specific categories of target states, such as stabilizer states or entangled states [126], although there exist methods that are applicable to general quantum states [187].

In this chapter, however, we are interested in the more specific task of *fidelity estimation*, in which one directly estimates the fidelity between the experimentally acquired state and the target state as a measure of distance between them. We are further interested in methods which provide some rigorous measure of confidence in the estimate. Prototypical among such methods is the eponymous Direct Fidelity Estimation (DFE) technique [50, 37], which can be used to rigorously compute a confidence interval on the true fidelity without needing to perform the full tomography, and degrades slowly as one takes fewer measurements. However, the technique requires the use of a pre-determined experimental protocol, which may be unnecessarily restrictive in practice. Indeed, it is often preferable in an experimental context to determine such quantities of interest with an entirely arbitrary sequence of measurements, particularly if this sequence can be determined prior to the collection of any data. In particular, one might hope to find the most informative sequence of measurements, which is the focus of this chapter.

3.3 Background and Related Work

For the remainder of this chapter, we consider a quantum system with a d-dimensional Hilbert space over $\mathbb{C}$. We identify each state in this system with a density matrix, a $d \times d$ positive semidefinite matrix of unit trace, and we denote the set of such density matrices as $\mathcal{X}$. When the quantum state is *pure*, the associated density matrix is additionally rank one [119]. Otherwise, the state is *mixed*. We consider in this chapter only scenarios in which the target state ρ is pure, as pure states are fundamental to the theoretical framework of quantum mechanics, and in particular quantum computing [50]. However, the unknown, experimentally constructed state σ is almost necessarily mixed due to imperfections in the creation process.

When the target state is pure, the fidelity between the two is a linear function of σ for a fixed ρ, defined by the inner product of the Hilbert space

$$F(\sigma; \rho) = \langle \sigma, \rho \rangle = \mathrm{Tr}(\sigma\rho). \tag{3.1}$$

Because the fidelity is dependent on the unknown state, which may differ considerably from the

expected target, we can only estimate it experimentally based on a set of observations from a particular measurement protocol. Each of the L measurement settings of the protocol is described by a distinct positive operator-valued measure (POVM) $\{E_1^{(\ell)}, \dots E_{N_\ell}^{(\ell)}\}$, a set of positive semidefinite operators that sum to the identity. When the experimental state σ is observed via the ℓ^{th} measurement setting, there are N_ℓ possible outcomes, mapped to by $k \in \{1, \dots, N_\ell\}$, with each outcome corresponding to a single element of the POVM $E_k^{(\ell)}$. This correspondence is described by Born's rule, which states that the probability of observing outcome k in the ℓ^{th} measurement setting is given by $p_\sigma^{(\ell)}(k) = \text{Tr}(E_k^{(\ell)}\sigma)$. Each measurement, or *shot*, is repeated S_ℓ times to produce independently and identically distributed outcomes $\{o_1^{(\ell)}, \dots, o_{S_\ell}^{(\ell)}\}$.

There is much flexibility in the choice of POVM, each corresponding to different experimental setups. For example, a measurement settings that is particularly amenable to theoretical evaluation is described by the POVM $\{\rho, I - \rho\}$, where ρ is itself the target state. For such a POVM, the fidelity is naturally equal to the expected value of the outcome of the single measurement, as

$$p_\sigma^1(1) = \text{Tr}(\rho\sigma) = F(\sigma; \rho). \tag{3.2}$$

However, it is unlikely that an experimental setup would be able to perform such a measurement directly [152].

Instead, the most common type of measurement are those related to the set of $4^d - 1$ Pauli observables. For a single qubit system, the Pauli observables are described by the set of 2×2 matrices

$$\sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{3.3}$$

to which we add the 2×2 identity matrix σ_I to form a complete basis. To consider higher dimensional systems, we simply take tensor products of these matrices to form a full set of 4^d Pauli observables $\{W_\ell\}_{\ell=1}^{4^d}$, often excluding the $d \times d$ identity matrix as being inherently uninformative. Each W_ℓ is positive semi-definite, traceless, and the eigenvalues lie in the set $\{\pm1\}$.

Each Pauli observable can be used to construct different types of POVM. For example, each

element of the POVM associated with the observable W_i can be constructed as a projector onto each of its two eigenspaces (for eigenvalues ± 1), giving

$$\left\{E_k^{(\ell)}\right\}_{k=1}^{N_\ell=2} = \left\{\tfrac{1}{2}(I+W_\ell), \tfrac{1}{2}(I-W_\ell)\right\} = \left\{E_+^{(\ell)}, E_-^{(\ell)}\right\}.$$

We refer to such POVMs as "coarse", in the sense that they only have $N_\ell = 2$ outcomes for arbitrary dimension.

POVMs of this type are particularly theoretically easy to work with. By assigning to each outcome a value in $o_k^{(\ell)} \in \{-1, +1\}$, the expected value of the corresponding measurement of σ is equal to the fidelity between σ and the observable W_ℓ:

$$\begin{aligned}
\mathbb{E}[o_k^{(\ell)}] &= +1 \cdot p_\sigma^+(+1) + (-1) \cdot p_\sigma^-(-1) \\
&= \mathrm{Tr}(E_+^{(\ell)}\sigma) - \mathrm{Tr}(E_-^{(\ell)}\sigma) \\
&= \mathrm{Tr}\left(\tfrac{1}{2}(I+W_\ell)\sigma\right) - \mathrm{Tr}\left(\tfrac{1}{2}(I-W_\ell)\sigma\right) \\
&= \mathrm{Tr}(W_\ell\sigma).
\end{aligned} \tag{3.4}$$

If measurements in all $4^d - 1$ POVMs are taken, then because the observables W_ℓ form a complete basis, both a full tomographic reconstruction of σ and an estimate of the fidelity can be reconstructed from the expected values of these coarse measurements.

However, individual measurements of this kind are not as informative as certain hardware configurations permit in practice, requiring more measurements to find accurate estimates of each Pauli observable. This is particularly significant if fewer than all $4^d - 1$ POVMs are measured. An alternative POVM that is, in some sense, more "expressive", is one that projects onto the full eigenspace of the observable W_k for a total of $N_\ell = 2^d$ outcomes. Elements of such a POVM are constructed through tensor products of the projectors onto the eigenspaces of the Pauli observables $\sigma_X, \sigma_Y, \sigma_Z$, equivalent to the coarse POVM of a single-qubit system. We refer to measurements of this type as "fine". As an example, consider the observable $W_k = \sigma_X \otimes \sigma_Y \otimes \sigma_Z$ in a three-qubit system. Each single qubit operator σ_k has two projectors $E_+^{(k)}$ and $E_-^{(k)}$, and tensor products across

each qubit gives a total of $2^3 = 8$ projectors for the observable W_k:

$$\begin{aligned}\{E_k^{(\ell)}\}_{k=1}^{N_\ell=8} = \big\{&E_+^{(X)} \otimes E_+^{(Y)} \otimes E_+^{(Z)}, \quad E_+^{(X)} \otimes E_+^{(Y)} \otimes E_-^{(Z)}, \\ &E_+^{(X)} \otimes E_-^{(Y)} \otimes E_+^{(Z)}, \quad E_+^{(X)} \otimes E_-^{(Y)} \otimes E_-^{(Z)}, \\ &E_-^{(X)} \otimes E_+^{(Y)} \otimes E_+^{(Z)}, \quad E_-^{(X)} \otimes E_+^{(Y)} \otimes E_-^{(Z)}, \\ &E_-^{(X)} \otimes E_-^{(Y)} \otimes E_+^{(Z)}, \quad E_-^{(X)} \otimes E_-^{(Y)} \otimes E_-^{(Z)}\big\}.\end{aligned} \tag{3.5}$$

For a given Pauli observable, each individual fine POVM is more informative than the coarse equivalent. However, this particular construction means introduces ambiguities between the POVMs for different Pauli observables. Because the basis operator σ_I is equal to the identity matrix, it cannot provide information about the corresponding component state of the system. On the other hand, it must be possible for the experimental system which is modeled by the POVMs to measure *something* about this component. Without loss of generality, we choose to model this ambiguity through redundancy with the σ_Z basis operator (since σ_I and σ_Z share the same eigenvectors), simply setting $E_+^{(I)} = E_+^{(Z)}$ and $E_-^{(I)} = E_-^{(Z)}$, since otherwise $E_+^{(I)} = I$ and $E_-^{(I)} = 0$. Under this assumption, the total number of measurement settings described by fine POVMs is 3^d as, for example, the observable $W_k = \sigma_X \otimes \sigma_Y \otimes \sigma_Z$ is described by the same fine POVM as $W_{k'} = \sigma_X \otimes \sigma_Y \otimes \sigma_I$. This is in contrast to the corresponding coarse POVMs, which are distinct by construction:

$$\begin{aligned}\left\{E_k^{(XYZ)}\right\}_{k=1}^{N_\ell=2} &= \left\{\tfrac{1}{2}(I + \sigma_X \otimes \sigma_Y \otimes \sigma_Z), \tfrac{1}{2}(I - \sigma_X \otimes \sigma_Y \otimes \sigma_Z)\right\}, \\ \left\{E_k^{(XYI)}\right\}_{k=1}^{N_\ell=2} &= \left\{\tfrac{1}{2}(I + \sigma_X \otimes \sigma_Y \otimes \sigma_I), \tfrac{1}{2}(I - \sigma_X \otimes \sigma_Y \otimes \sigma_I)\right\}.\end{aligned} \tag{3.6}$$

Although these fine measurements are fundamentally more informative, many methods of analysis instead consider only measurements with a coarser set of 2 outcomes, which can be more tractable for statistical analysis. This is the case for the Direct Fidelity Estimation (DFE) method, in which Pauli measurements are selected randomly according to an "importance-weighting" rule. Given target state ρ, the Pauli observable W_k is selected with probability

$$\Pr(k) = \mathrm{Tr}\left(\rho W_k/\sqrt{d}\right)^2, \tag{3.7}$$

and is measured according to the corresponding coarse POVM. An estimate of the fidelity is constructed as the weighted average of the observed outcomes, along with a confidence interval derived from Chebyshev's inequality [50]. An important feature of this method is that it is prescriptive, and describes a specific measurement protocol of Pauli observables that must be considered through the described coarse POVM.

Other methods of experiment design for quantum state verification are more agnostic to the details of the measurement scheme. For example, the work of [98] describes a strategy for ordering measurement settings according to measured expected values of the observables themselves, rather than individual outcomes. This motive is consistent with recent work by [136], which suggests that the total number of measurement settings used in a tomographic reconstruction is more important to the quality of the numerical result than the number of observations made per setting, suggesting that even poor estimates of each observable can be compensated for by a large number of different settings.

In any experimental setting, each measurement of σ is performed on a different copy of the state, destroying it in the process, and requiring the experimenter to create another duplicate state. It is for this reason that a critical component of any fidelity estimation procedure is to perform an accurate estimation with as few measurements as possible, as each one costs the experimenter time. At the same time, each additional measurement **setting** considered incurs a greater additional cost than repetition of a single observation due to the nature of the experiment. The work of [187] describes a method of direct fidelity estimation from very few different measurement schemes by means of a neural network. The input of the described network is the observed experimental outcomes from a fine POVM, and the output is different fidelity intervals, in effect providing a lower bound on the fidelity. While this bound is computed from very few different measurement settings, doing so for a new target state requires fully re-training the network, limiting the applicability of such an approach to arbitrary target states. Furthermore, the method lacks rigorous guarantees on either the estimate or on uncertainty of the estimate.

An important feature of our proposed experimental design framework is that, in contrast

to these methods, it is highly flexible to the choice of measurement protocol (both in terms of the selection of measurement settings and the number of outcomes per setting) while still taking full advantage of more informative individual measurements. Furthermore, the method can be performed prior to the collection of any data, as it computes a bound on the fidelity in the worst case, which is the key feature that will allow us to compute the most informative measurements.

3.4 Methods

A theoretical framework for fidelity estimation that can be applied to an arbitrary experimental protocol is introduced in [152]. This technique, which we refer to as the *optimal minimax method* produces, both a single estimator $\widehat{F}$ of fidelity and a bound on its error $\widehat{R}$ in the worst case of σ. The estimator $\widehat{F}\left(\{o_i^{(\ell)}\}\right)$ is itself defined as an affine function of the observed outcomes from each measurement, mapping them to an estimate of the true fidelity F. Importantly, this estimator $\widehat{F}$ is constructed to have a near-optimal *risk* $R(\widehat{F})$ among all possible estimators of F, where $R(\widehat{F})$ is essentially the width of the smallest symmetric confidence interval around the estimate $\widehat{F}$ that contains F with a probability greater than $1-\delta$. The value of $\widehat{R}$ is itself constructed as an upper bound of its true, and unknown risk $R(\widehat{F})$, although the method of its construction ensures that the difference between these two values is reasonable for confidence levels above 90%. Nevertheless, we will in this text refer to $\widehat{R}$ simply as "the minimax risk" for simplicity, recognizing that it also defines the width of a confidence interval around $\widehat{F}$ which contains F with probability greater than $1-\delta$ [86, 153].

In practical implementation, the minimax risk $\widehat{R}$ can be computed independently of, and indeed prior to, the estimator $\widehat{F}$. Specifically, it is defined as the optimal value of the saddle point optimization problem:

$$\widehat{\mathcal{R}} = \inf_{\alpha>0}\left\{\alpha\ln(2/\delta) + \max_{\chi_1,\chi_2\in\mathcal{X}}\left[\frac{1}{2}\langle\rho,\chi_1\rangle - \frac{1}{2}\langle\rho,\chi_2\rangle + \alpha\ln(\mathrm{AffH}(A(\chi_1),A(\chi_2)))\right]\right\}, \tag{3.8}$$

where $\mathrm{AffH}(A(\chi_1), A(\chi_2))$ is the Hellinger affinity, a jointly log-concave function defined by

$$\ln\left(\mathrm{AffH}(A(\chi_1), A(\chi_2))\right) = \sum_{\ell=1}^{L} \frac{S_\ell}{2} \ln\left[\sum_{k=1}^{N_\ell} p_{\chi_1}^{(\ell)}(k) p_{\chi_2}^{(\ell)}(k)\right]. \tag{3.9}$$

Furthermore, the ability of this technique to compute an error bound in the worst case means that both it and the estimator can be computed prior to not only the estimator, but even the collection of any data. We leverage this fact to select an experimental protocol that minimizes the calculated risk $\widehat{R}$. To do so rigorously, we consider the $\widehat{\mathcal{R}}$ as a function of the experimental protocol, and design a measurement scheme that minimizes this value, subject to a budget constraint on the total number of shots.

Our goal is to allocate a fixed budget of S_{tot} observations, described by a vector of shot counts $\vec{S} = (S_1, \ldots, S_L)^T$ for each of a fixed set of L available measurement settings, in a way that minimizes the risk in Equation (3.8):

$$\begin{aligned} \underset{\vec{S}}{\text{minimize}} \quad & \widehat{\mathcal{R}}(\vec{S}), \\ \text{subject to} \quad & \sum_{\ell=1}^{L} S_\ell \leq S_{\mathrm{tot}}. \end{aligned} \tag{3.10}$$

Since $\widehat{\mathcal{R}}(\vec{S})$ itself is the solution to an optimization problem—which we refer to as the "inner problem"—this is a nested optimization problem. Unfortunately, Equation (3.10) is difficult to solve since it is both integer-valued and non-convex.

To address the former concern, we first note that although the S_ℓ are in principle integer-valued, with each representing discrete measurements of the experimental state, the objective function $\widehat{\mathcal{R}}$ allows for a real-valued treatment of these arguments. This means that during minimization of the risk, we can allow each S_ℓ to take non-integer values, and simply round them at the end, which typically has small effect since S_{tot} is typically on the order of 100s or 1000s, or even larger.

This relaxation does not guarantee a global minimizer, and indeed finding a true global minimizer is an intractable combinatorial problem. However, our goal is merely to find a *good enough* protocol, and in particular anything that improves upon the conventional alternative (such as that of DFE or using uniformly distributed shot counts).

To solve the outer non-convex optimization problem, we employ the `scipy.optimize` library, utilizing the constrained trust-region methods within. This, in turn, requires efficient evaluation of the risk function $\widehat{\mathcal{R}}$ and its gradient with respect to the vector of shot counts $\vec{S}$. By inspection, the objective function $\widehat{\mathcal{R}}$ is fully continuous and differentiable with respect to the vector $\vec{S}$ for non-zero shot counts. Using Equation (3.8), we derive

$$\nabla_{\vec{S}}\widehat{\mathcal{R}}_\ell(\vec{S}) = \alpha^* \ln\left(\sum_{k=1}^{N_\ell}\sqrt{p^{(\ell)}_{\chi_1^*}(k)p^{(\ell)}_{\chi_2^*}(k)}\right), \tag{3.11}$$

where α^*, χ_1^*, and χ_2^* are the optimal values of the saddle point problem in Equation (3.8) at the input vector $\vec{S}$. Importantly, if the objective function has already been evaluated at this input vector, then constructing the gradient comes at essentially no additional cost since α^*, χ_1^*, and χ_2^* are already known.

To do so, we solve this inner optimization problem for a fixed experimental protocol using the standard convex optimization software `cvxpy`. This requires rewriting Equation (3.8) in the standard form of a convex optimization problem,

$$\begin{aligned} \underset{\chi_1,\chi_2\,\in\,\mathcal{X}}{\text{minimize}} \quad & \langle \rho, \chi_2 - \chi_1\rangle, \\ \text{subject to} \quad & 2\ln(\epsilon/2) - 2\ln(\text{AffH}(A(\chi_1), A(\chi_2))) \leq 0. \end{aligned} \tag{3.12}$$

While solving this problem results in both the risk $\widehat{R}$ and optimal values of χ_1^* and χ_2^*, the construction of both the estimator $\widehat{F}$ and the gradient $\nabla_{\vec{S}}\widehat{\mathcal{R}}_\ell(\vec{S})$ requires the optimal value α^* in the original saddle-point formulation as well. To obtain this value without solving the saddle-point problem directly, we show that this parameter α is actually the dual variable associated with the constraint on the Hellinger affinity.

Taking p^* as the primal optimal value of Equation (3.12), p^* is related to the risk $\widehat{R}$ by $-\frac{1}{2}p^* = \widehat{R}$. Introducing $\widetilde{\alpha}$ as a dual variable, the Lagrangian for our nonlinear programming problem is given by

$$\begin{aligned} \mathcal{L}(\chi_1, \chi_2; \widetilde{\alpha}) = \langle \rho, \chi_2 - \chi_1\rangle \\ + \widetilde{\alpha}\Big(2\ln(\epsilon/2) - 2\ln(\text{AffH}(A(\chi_1), A(\chi_2)))\Big). \end{aligned} \tag{3.13}$$

This defines a concave Lagrange dual function $g : \mathbb{R} \to \mathbb{R}$ defined by

$$g(\widetilde{\alpha}) = \min_{\chi_1, \chi_2 \in \mathcal{X}} \mathcal{L}(\chi_1, \chi_2; \widetilde{\alpha}). \tag{3.14}$$

Let d^* denote the maximum value of the function g. Because the problem in Equation (3.12) is convex and Slater's condition holds (as strict feasibility is achieved whenever $\chi_1 = \chi_2$), we have strong duality between the two problems. This means that $p^* = d^*$, and it follows that

$$\begin{aligned} 2\widehat{R} = -p^* = -d^* &= -\max_{\widetilde{\alpha} \geq 0} g(\widetilde{\alpha}) \\ &= \min_{\widetilde{\alpha} \geq 0} \max_{\chi_1, \chi_2 \in \mathcal{X}} -\mathcal{L}(\chi_1, \chi_2; \widetilde{\alpha}) \\ &= \min_{\widetilde{\alpha} \geq 0} \max_{\chi_1, \chi_1 \in \mathcal{X}} \Bigg\{ \langle \rho, \chi_1 - \chi_2 \rangle \\ &\qquad + 2\widetilde{\alpha} \Big[\ln(\mathrm{AffH}(A(\chi_1), A(\chi_2))) - \ln(\epsilon/2) \Big] \Bigg\} \\ &= \inf_{\widetilde{\alpha} > 0} \Bigg\{ 2\widetilde{\alpha} \ln(2/\epsilon) + \max_{\chi_1, \chi_2 \in \mathcal{X}} \Big[\langle \rho, \chi_1 \rangle - \langle \rho, \chi_2 \rangle \\ &\qquad + 2\widetilde{\alpha} \ln(\mathrm{AffH}(A(\chi_1), A(\chi_2))) \Big] \Bigg\}. \end{aligned} \tag{3.15}$$

With $\widetilde{\alpha} = \alpha$, this is exactly the expression in Equation (3.8), showing equivalence of the two values. Importantly, this means that the estimator found by solving the saddle-point problem in Equation (3.8) can be equivalently constructed by finding both primal and dual optima of Equation (3.12), which can be readily obtained through the solution with `cvxpy`.

3.5 Results

Given a set of available measurement settings, our method for optimal experiment design is capable of producing tighter confidence intervals than conventional alternatives.

3.5.1 Comparison to Uniform and DFE-derived Allocations

As our most direct comparison, we consider alternative measurement allocations to ones generated by our optimization procedure. We take as our first point of comparison a common approach in which the budget is uniformly distributed among all available non-trivial measurements [136, 96].

We also compare to a more sophisticated technique, a practical variant of the previously described DFE technique. In the original DFE method, "coarse" measurements corresponding to the Pauli observables are selected at random according to a prescribed importance sampling rule. Because this strategy is necessarily probabilistic, the exact allocation of measurements cannot be known in advance, but a benefit of this probabilistic approach is that the method comes with a rigorous confidence interval. To facilitate comparison with our scheme, we make the following modification of DFE. We consider a "DFE-derived" protocol to be one that deterministically distributes the measurement budget among Pauli measurements according to their relative weighting in the relevant sampling rule, $\Pr(k) = [\mathrm{Tr}(\rho W_k)]^2$. An exception to this allocation strategy is made with the identity Pauli operator W_0, as the associated measurement can be done "by hand", in a sense, and should therefore not contribute to the budget.

To estimate the risk of any given protocol (found either by our method or by DFE or by uniform measurements), we always use the *minimax optimal* confidence interval. Doing so provides a uniform comparison between all three experimental protocols, establishing the same type of worst case bound for each method. While this does result in a different analysis of the DFE method than suggested by the original text [50], it is nevertheless a common practice in the contemporary fidelity estimation literature, as (See [187] as a practical example).

We consider the performance of these three types of experimental protocol across a number of comparative scenarios. In all cases, we compare the risk associated with a confidence level of $1 - \delta = 0.95$.

As a baseline numerical experiment, we compare the performance of each method on 3-qubit states, one with a GHZ stabilizer state, and the other a W state, each measured with a coarse POVM over a budget of $S_{\mathrm{tot}} = 630$ measurements. Importantly, we reiterate that the optimized experimental protocol, and all risk estimates, are computed without access to any information about the experimental state. We compile these results in Table 3.1, and see that our scheme provides a more efficient utilization of available experimental resources, providing clear improvement to the risk estimate using the optimized protocol over the uniform protocol.

This scenario, in which each of $4^3 - 1$ measurements have only two possible outcomes, is the most similar to the typical application of the DFE method, and as such, we see less distinction between the optimized protocol and the DFE-derived protocol. Indeed in both cases, we see that the selected measurements are the same between the two methods, and only for the W state case do we see a difference in the number of shots allocated to each measurement, leading to a slight improvement in the risk estimate. Nevertheless, the fact that the proposed optimized is able to accurately recreate the DFE-derived protocol is notable, as the optimization procedure itself is initialized with the uniform protocol.

For a more complex example, we consider the same types of quantum states, but optimize the budget of S_{tot} shots over the set of 3^3 fine measurements, each with 2^3 outcomes. As described previously, there are fewer settings for these fine measurements than in the coarse alternative, as we model measurements with uninformative components as POVMs with components which are redundant in the Z basis. However, even if the fine POVMs associated with, for example, Pauli observables W_{IXZ} and W_{ZXZ} are equivalent, the importance sampling rule which defines the DFE protocol does distinguish them, i.e., $\text{Tr}\left(\rho W_{\text{IXZ}}\right) \neq \text{Tr}\left(\rho W_{\text{ZXZ}}\right)$, and allocates a different number of measurements to each. Maintaining a fair comparison between the DFE-derived experimental protocol and the proposed scheme requires resolving this ambiguity, which we choose to do by aggregating the shots allocated towards redundant measurements in our DFE-derived protocol. The exception to this aggregation is the Pauli identity observable W_{III}, which we ignore entirely, as such a measurement would never be carried out experimentally as already discussed.

As an aside, this is but one of many possible ways to resolve this ambiguity. For example, one could instead *average* the number of shots allocated towards redundant measurements. Or, one could consider only the non-redundant measurements which do not involve the identity operator. In our numerical experiments, we have observed only minimal differences between these three approaches, but it should be noted that very existence of this ambiguity demonstrates the difficulties of cleanly reconciling a fine POVM with a DFE-derived protocol, with there are currently few clear alternatives in the literature.

3-qubit GHZ State, Coarse Measurements, $S_{\text{tot}} = 630$

Measurement	Uniform Shots	DFE-Derived Shots	Optimized Shots
IZZ	10	90	90
XXX	10	90	90
XYY	10	90	90
YXY	10	90	90
YYX	10	90	90
ZIZ	10	90	90
ZZI	10	90	90
Remaining 56 Measurements	10	0	0
Risk $\widehat{R}$:	0.2723	0.0944	0.0944

3-qubit W State, Coarse Measurements, $S_{\text{tot}} = 630$

Measurement	Uniform Shots	DFE-Derived Shots	Optimized Shots
IIZ	10	10	19
IXX	10	40	38
IYY	10	40	38
IZI	10	10	19
IZZ	10	10	19
XIX	10	40	38
XXI	10	40	38
XXZ	10	40	38
XZX	10	40	38
YIY	10	40	38
YYI	10	40	38
YYZ	10	40	38
YZY	10	40	38
ZII	10	10	19
ZIZ	10	10	19
ZXX	10	40	38
ZYY	10	40	38
ZZI	10	10	19
ZZZ	10	90	57
Remaining 44 Measurements	10	0	0
Risk $\widehat{R}$:	0.2786	0.1552	0.1486

Table 3.1: Conventional vs. optimized protocols using coarse POVMs on 3-qubit GHZ and W states. We can wee that our scheme provides a more efficient utilization of available experimental resources, providing a lower value for the risk using the same volume of measurements. In this example, we can see that the comparative effect increases with the complexity of the target state.

Because this is a scenario in which the traditional DFE method cannot not apply directly, we see in Table 3.2 more pronounced improvement in the risk estimate using the optimized protocol, particularly in the more complex W state. Furthermore, the optimized protocol reveals additional measurements that may not be suggested by any underlying theory, as seen in the W state case.

In the final comparison, we consider a 5-qubit W state, again measured with a fine POVM. It is in this example that we see the greatest comparative improvement among the three methods. This is partially because as the size of the system increases, the fine POVM becomes more expressive relative to the coarser alternative. In contrast to previous examples with the W state, there is a greater degree of non-uniformity of the optimized protocol, as many measurements are only allocated a single shot. This, along with the observation that the uniform allocation outperforms the typically more sophisticated DFE-derived protocol, is consistent with current literature that states that protocols with additional measurement settings often outperform more accurate estimates of individual observables [136]. At the same time, there is still a clear advantage towards allocating the majority of measurements towards a handful of particularly informative settings, even if these settings are not considered "important" by the weighting function of the DFE method. Furthermore, this is also a scenario for which the proposed optimization framework is particularly well-suited, as the size of the optimization problem is considerably smaller in the case of a fine POVMs, as there are altogether fewer kinds of measurements.

3.5.2 Comparison Across Noisy States

The above numerical experiment shows improvement in terms of the calculated minimax risk, which is a **bound** on the estimation error when applied to measured outcomes, and as a result does not consider the experimental state σ. However, we can verify these bounds empirically by performing repeated numerical simulations of observed measurements and using each to evaluate an estimate of the fidelity. In doing so, we demonstrate that not only does the optimization procedure improve the risk produced by the minimax procedure, but consequently improves the actual estimates of fidelity.

3-qubit GHZ State, Fine Measurements, $S_{\text{tot}} = 630$

Measurement	Uniform Shots	DFE-Derived Shots	Optimized Shots
XXX	23	90	105
XYY	23	90	105
YXY	23	90	105
YYX	23	90	105
ZZZ	23	270	210
Remaining 22 Measurements	23	0	0
Risk $\widehat{R}$:	0.2615	0.0823	0.0809

3-qubit W State, Fine Measurements, $S_{\text{tot}} = 630$

Measurement	Uniform Shots	DFE-Derived Shots	Optimized Shots
XXX	23	0	81
XXZ	23	70	61
XZX	23	70	61
YYY	23	0	81
YYZ	23	70	61
YZY	23	70	61
ZXX	23	70	61
ZYY	23	70	61
ZZZ	23	210	104
Remaining 18 Measurements	23	0	0
Risk $\widehat{R}$:	0.1582	0.1279	0.1041

Table 3.2: Conventional vs. optimized protocols using fine POVMs on 3-qubit GHZ and W states. We repeat the same example with a more expressive set of measurements, and see that our optimized protocol becomes more advantageous as the complexity of the problem increases.

5-qubit W State, Fine Measurements, $S_{\text{tot}} = 2430$

Measurement	Uniform Shots	DFE-Derived Shots	Optimized Shots	Measurement	Uniform Shots	DFE-Derived Shots	Optimized Shots
XXXXZ	10	0	20	YYYZZ	10	0	82
XXXYY	10	0	1	YYZYY	10	0	20
XXXZX	10	0	20	YYZYZ	10	0	82
XXZZZ	10	100	22	YZZYZ	10	100	22
XXXZZ	10	0	82	YYZZY	10	0	82
XXYXY	10	0	1	YYZZZ	10	97	22
XXYYX	10	0	1	YZYYY	10	0	20
XXYYY	10	0	1	YZYYZ	10	0	82
XXZXX	10	0	20	YZYZY	10	0	82
XXZXZ	10	0	82	YZYZZ	10	97	22
XXZZX	10	0	82	YZZYY	10	0	82
XYXXY	10	0	1	YZZZY	10	100	22
XYXYX	10	0	1	ZXXXX	10	0	20
XYXYY	10	0	1	ZXXXZ	10	0	82
XYYXX	10	0	1	ZXXZX	10	0	82
XYYXY	10	0	1	ZXXZZ	10	100	22
XYYYX	10	0	1	ZXZXX	10	0	82
XZXXX	10	0	20	ZXZXZ	10	100	22
XZXXZ	10	0	82	ZXZZX	10	100	22
XZXZX	10	0	82	ZYYYY	10	0	20
XZXZZ	10	100	22	ZYYYZ	10	0	82
XZZXX	10	0	82	ZYYZY	10	0	82
XZZXZ	10	100	22	ZYYZZ	10	100	22
XZZZX	10	100	22	ZYZYY	10	0	82
YXXXY	10	0	1	ZYZYZ	10	100	22
YXXYX	10	0	1	ZYZZY	10	100	22
YXXYY	10	0	1	ZZXXX	10	0	82
YXYXX	10	0	1	ZZXXZ	10	100	22
YXYXY	10	0	1	ZZXZX	10	100	22
YXYYX	10	0	1	ZZYYY	10	0	82
YYXXX	10	0	1	ZZYYZ	10	100	22
YYXXY	10	0	1	ZZYZY	10	100	22
YYXYX	10	0	1	ZZZXX	10	100	22
YYYXX	10	0	1	ZZZYY	10	100	22
YYYYZ	10	0	20	ZZZZZ	10	423	110
YYYZY	10	0	20	172 Other Measurements	10	0	0

	Uniform	DFE-Derived	Optimized
Risk $\widehat{R}$:	0.0986	0.1128	0.0711

Table 3.3: Conventional vs. optimized protocols using fine POVMs on a 5-qubit W state. We see that our optimized protocol becomes more advantageous as the complexity of the measurement scheme increases, and that even the uniform allocation outperforms the DFE-based protocol. We also observe that the optimized protocol is less intuitive when fine measurements are used in the larger problem.

These simulations are performed on a pure and totally random 3-qubit state to which varying levels of depolarizing noise are added. As before, we consider the risk associated with a confidence level of 0.95. For each level of depolarizing noise, we simulate 5000 sets of measured data, using a budget of $S_{\text{tot}} = 1250$ shots for each instance. This budget was selected to ensure that the risk is less than 0.10 in the optimized case.

For each level of depolarizing noise in Figure 3.1, we plot a distribution of measured values for the fidelity along with the true fidelity with the target state. For each experimental protocol (DFE-derived in blue and optimized in orange) we plot a minimax confidence interval around the true fidelity, as well as an empirically constructed confidence interval that is selected to have minimum width while capturing 95% of the estimates. As expected, the optimized protocol both shortens the length of the minimax confidence interval and improves the estimates of the fidelity. We make particular note of the fact that because the calculated minimax risk is minimized for a given target state with no reference to observation data, the width of the minimax confidence interval is equal across the noise levels.

3.6 Discussion

3.6.1 Construction of DFE-Adversarial Quantum State

The experimental protocol suggested by the DFE method is commonly used, but as we have seen in our numerical results, can underperform when used alongside other methods of statistical analysis. However, it is also useful to consider a target state which in some sense has worst-case performance when measured with the DFE analysis, but becomes typical under our optimized protocol.

To construct this adversarial state, we first consider a 3-qubit pure state ρ_1 which is strongly aligned in the Z basis, in the sense that the DFE importance weighting distribution $\text{Tr}\left(\rho W_k/\sqrt{d}\right)^2$ is nonzero only for the Pauli observables IIZ, IZI, IZZ, ZII, ZIZ, ZZI, and ZZZ. While these 8 Pauli observables generate distinct *coarse* POVMs, the corresponding fine POVMs are

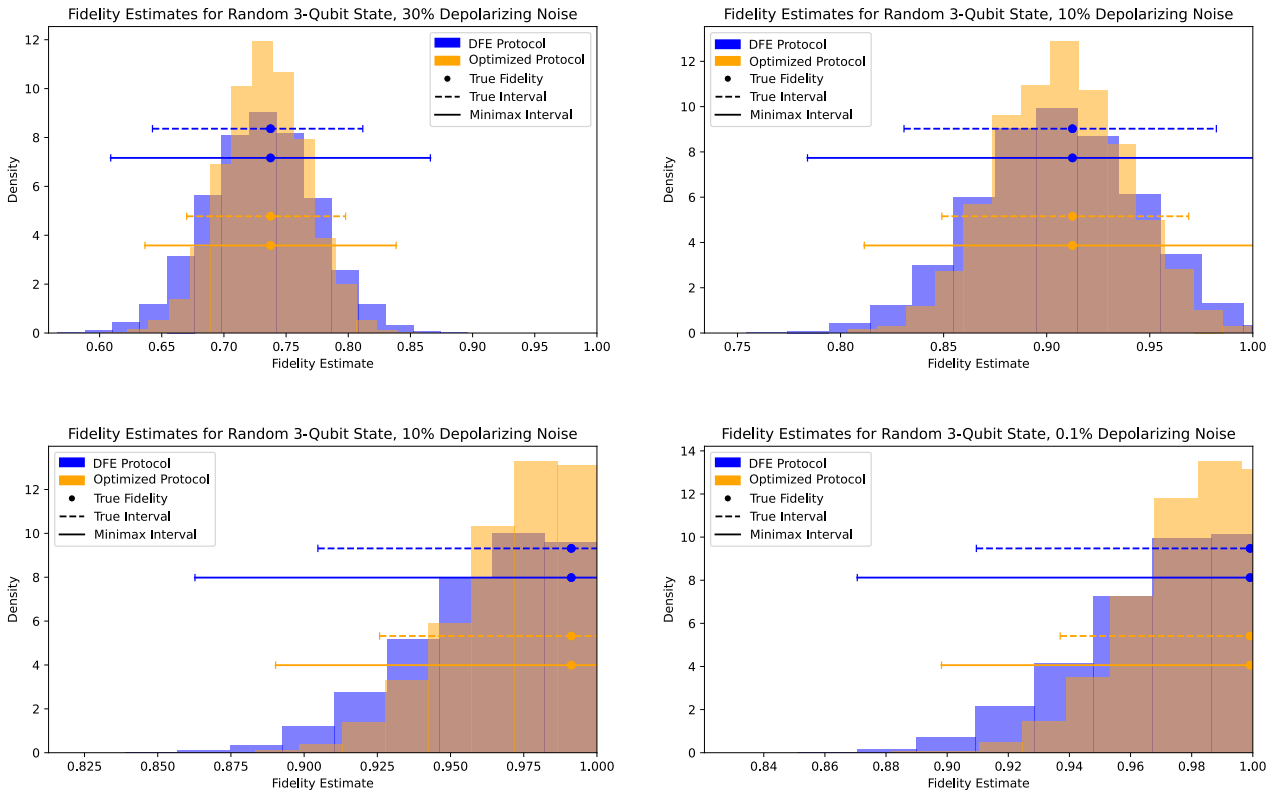


Figure 3.1: Simulated fidelity estimates across noise levels compared to a DFE-based protocol (orange). Across all noise levels, we see that the optimized experimental protocol (blue) shows clear improvements in both the width of the interval and the fidelity estimates themselves.

not distinguishable (assuming the experimental system is similarly aligned in the Z basis). In effect, a state with this structure would correspond to a DFE protocol which (correctly) allocates measurements towards the single POVM for Pauli observable ZZZ.

We then consider a second 3-qubit pure state ρ_2 whose DFE importance weighting is more evenly distributed across all 64 Pauli observables, which we construct as an entirely pure random state. We then combine these two states to form ρ_3, which will continue to have W_{ZZZ} as the "most important" observable, but has a nonzero weighting for the remaining observables as well.

To construct the adversarial state ρ_3, it is important that we not simply define ρ_3 as a convex combination of ρ_1 and ρ_2, as this would result in a "mixed" state which is not pure. Instead, we first define $\rho_1 = |\psi_1\rangle\langle\psi_1|$ and $\rho_2 = |\psi_2\rangle\langle\psi_2|$, where $|\psi_1\rangle$ and $|\psi_2\rangle$ are state vectors in the Hilbert

space of the 3-qubit system. With this, we can construct $|\psi_3\rangle = \sqrt{p}|\psi_1\rangle + \sqrt{1-p}|\psi_2\rangle$ for some $p \in [0,1]$ (we take $p = 0.75$ in this example) followed by $\rho_3 = |\psi_3\rangle\langle\psi_3|$. This ensures that the "combined" (but not mixed) state ρ_3 is pure and has the desired properties.

We can now compare the risk produced by the minimax method on ρ_3 with the DFE protocol and our optimized protocol, demonstrating the poor performance of the minimax method when the majority of shots are allocated towards W_{ZZZ} as suggested by DFE. As seen in Figure 3.2, the protocol suggested by DFE proposes using over 4× as many measurements for the W_{ZZZ} POVM than the optimized protocol, which instead distributes shots much more evenly between measurements. Interestingly, the DFE protocol in this case performs even more poorly than a uniform protocol, which is not the case for a typical random 3-qubit state (see Table 3.2).

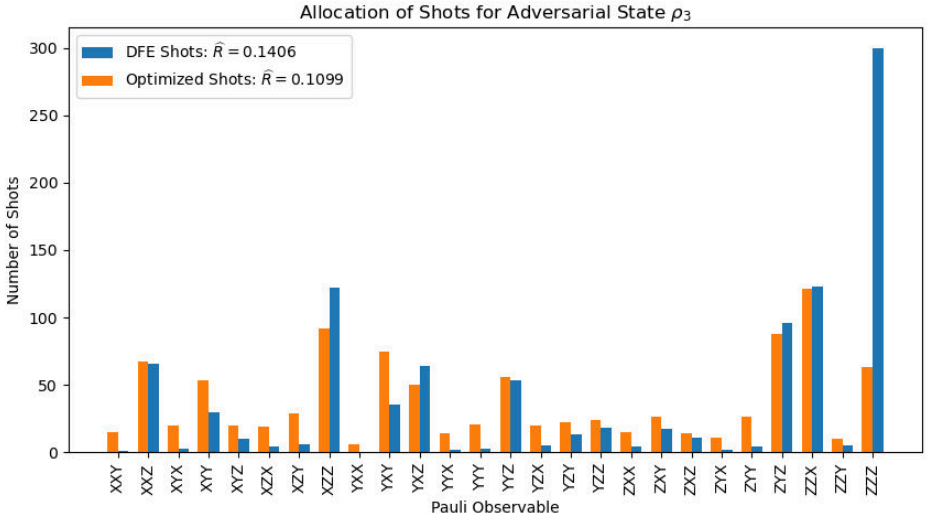


Figure 3.2: Comparison of DFE-based allocation and optimized allocation for an adversarial 3-qubit state. Although the DFE importance sampling would suggest that the Pauli observable W_{ZZZ} should be allocated the greatest number of measurements, the more performant optimized experimental protocol instead distributes shots much more evenly between measurements.

3.6.2 Optimal Experimental Protocol as an Importance Weighting

One of the key features of the DFE method is that the relative probability an individual Pauli observable being measured has utility beyond the fidelity estimate itself. In particular, the relevant

importance sampling rule is often used in other problem contexts [187], and being able to derive a "measurement order" for a given set of Pauli observables is useful more generally [98]. However, its use as an importance sampling rule is restricted to cases where the relevant Pauli observable is measured through a coarse POVM.

In contrast, we suggest (although do not prove rigorously) that the relative allocation of shots in an optimized experimental protocol, when generated over the full set of fine POVMs, can be used as a more sophisticated importance weighting in the context of the minimax fidelity estimation procedure. For the resulting allocation to truly define the most informative measurements for the fidelity of a fixed state ρ, one would have to show that taking the N measurements with the highest shot counts provides the lowest risk estimate of the fidelity among all other subsets of N measurements. Naturally, this is combinatorially infeasible for all values of N, and so we show this property empirically for $N = 3, 4, 5$ for a random 3-qubit target state ρ, for which there are a total of 3^3 possible fine POVM measurements.

Taking the adversarial state, we generate an optimal experimental protocol for the full set of measurement settings with a budget of $S_{\text{tot}} = 1000$ shots, and sort them in Table 3.4 according to the number of measurements allocated towards each. In Table 3.5, we consider all subsets of 3, 4, and 5 measurement settings, and present the five subsets of each cardinality which, when optimized with their own budget of 1000 shots, produces the lowest risk when evaluated by the minimax method. For each size of subset, we see that the subsets of measurements with the lowest risk estimates are indeed those which are allocated the most shots in the full experimental protocol. Interestingly, we observe that the individual measurements within these subsets are not necessarily allocated in the same way as in the original optimized protocol. In either case, however, this suggests that the settings which are allocated by the optimized experimental protocol are indeed the most informative for the fidelity estimate, and that for these small subsets of the original measurement settings, that our original allocation could indeed be considered as an importance weighting for the minimax method.

3-qubit Random State, Fine Measurements, $S_{\text{tot}} = 1000$

Measurement	Optimized Shots	Measurement	Optimized Shots
ZYX	115	XZZ	28
YYY	97	ZZY	24
ZXZ	78	XZY	23
XXX	75	XYY	22
ZZX	69	XZX	21
YYX	65	YXZ	18
XYZ	54	YZX	18
YZY	45	YXY	16
YXX	43	XXY	10
ZYY	39	ZXX	5
XXZ	34	XYX	4
ZXY	33	ZYZ	4
ZZZ	30	YYZ	0
YZZ	30		
Risk $\widehat{R}$: 0.1114			

Table 3.4: Optimized protocol for a pure 3-qubit target state considered in Table 3.5. Individual measurements are sorted by the number of shots allocated towards each.

3.6.3 Comparison to Maximum-Likelihood Estimation

The Maximum-Likelihood Estimation (MLE) method remains a common approach to fidelity estimation, despite the limitations of a more expensive reconstruction process and the lack of a rigorous error bound. The objective of the MLE method is reconstruction of an approximated experimental state $\widehat{\sigma}$ which is most likely to have generated the set of observations $\{o_k^{(\ell)}\}$. To simplify computation, we consider the collection of observations from the L^{th} measurement as a vector of frequencies $f_k^{(\ell)}$ for each of N_ℓ outcomes. In cases where $f_k^{(\ell)} = 0$, we apply a hedging procedure to improve the quality of the result [17]. Similarly, we consider the allocation of shots between measurement settings as a weighting w_ℓ which instead determines the *probability* of performing a measurement ℓ. With this framework, the log-likelihood function is given as

$$\log l(\sigma|f) = -\sum_{\ell=1}^{L}\sum_{j=1}^{N_\ell} w_\ell f_k^{(\ell)} \log\left(w_\ell\, p_\sigma^{(\ell)}(k)\right), \tag{3.16}$$

3 Measurement Subsets with Lowest Risk

Measurement Set	Shot Allocation	Risk $\widehat{R}$
YYY; ZXZ; ZYX	407; 276; 316	0.3743
XXX; YYY; ZYX	261; 414; 325	0.3856
YZY; ZXZ; ZYX	325; 249; 424	0.3882
YYX; YZY; ZXZ	335; 299; 364	0.4007
YYY; ZYX; ZZY	313; 435; 250	0.4017

4 Measurement Subsets with Lowest Risk

Measurement Set	Shot Allocation	Risk $\widehat{R}$
XXX; YYY; ZXZ; ZYX	183; 334; 178; 303	0.3290
YYY; ZXZ; ZYX; ZZY	325; 201; 317; 155	0.3326
YYY; ZXZ; ZYX; ZZX	354; 229; 227; 188	0.3353
XXY; YYY; ZXZ; ZYX	154; 307; 204; 333	0.3398
YYY; YZY; ZXZ; ZYX	202; 242; 261; 293	0.3408

5 Measurement Subsets with Lowest Risk

Measurement Set	Shot Allocation	Risk $\widehat{R}$
XXX; YYY; ZXZ; ZYX; ZZX	172; 277; 152; 254; 143	0.2934
XXX; YYY; ZXZ; ZYX; ZZY	185; 272; 117; 274; 150	0.2972
YYY; YZX; ZXZ; ZYX; ZZX	284; 161; 187; 227; 138	0.2997
XXX; YYY; YZX; ZXZ; ZYX	121; 273; 141; 218; 244	0.3003
XXX; YYY; YZY; ZXZ; ZYX	152; 267; 132; 187; 260	0.3013

Table 3.5: Subsets of measurements which, when optimized, produce the lowest risk for the state described in Table 3.4. Optimized expeerimental protocols for each subset are generated with a budget of $S_{\text{tot}} = 1000$ measurements. Note that in all three presented cases, the measurement subset with the lowest risk uses the same settings as those with the greatest allocation in the fully optimized experimental protocol, although we note that in general, the allocation *within* the subset does not match the allocation in the full protocol.

and we take $\widehat{\sigma} = \arg\min_{\sigma} \log l(\sigma|f)$. Of note is that this procedure is performed without reference to the target state ρ. This estimate can be used immediately to define a measure of the fidelity between the target state ρ and the approximated experimental state $\widehat{\sigma}$, given by $\widehat{F} = \mathrm{Tr}(\widehat{\sigma}\rho)$. To calculate the fidelity estimate with MLE for a given experimental protocol then, we first simulate a set of outcomes for the measurement scheme, reconstruct a tomographic approximation of the target state according to the above minimization problem, and then compute the fidelity between the target state and the reconstructed state. The optimization procedure which generates this approximation is implemented in `cvxpy`.

In principle, the MLE-based fidelity estimation procedure can be applied to any measurement protocol, even if they are most commonly applied to those with a uniform allocation of measurements. A natural question, then, is how an optimized experimental protocol performs compared to conventional alternatives when estimates are created with this MLE-based analysis, rather than the minimax analysis for which it was optimized.

In the following numerical example, we measure our adversarial state defined in Section 3.6.1 by $S_{\mathrm{tot}} = 1250$ observations via fine POVMs, for which the experimental state σ is generated by adding 10% depolatizing noise to ρ. By considering this adversarial state, we effectively maximize the performance difference between our optimized protocol and conventional alternatives, which in principle should highlight the effect of our optimization procedure on the MLE performance as well.

We reiterate that this MLE procedure does not inherently provide an error bound on the fidelity estimate. To provide a reasonable comparison to the minimax method, we use the common approach of Monte-Carlo resampling to compute some measure of uncertainty in the MLE fidelity estimate. This involves taking the original set of outcomes which were used to reconstruct the state, and using the relative frequency of each to resample a new set of "synthetic" outcomes. These synthetic outcomes are then used in a new MLE reconstruction, and therefore a new fidelity estimate. Naturally, this relative frequency may diverge considerably from the ground-truth distribution, which may introduce considerable bias in ensuing estimates. Nevertheless, we repeat this

resampling process $N = 5000$ times, from which we can generate empirical confidence intervals of the original fidelity estimate.

We first consider a DFE-based protocol, and compare in Figure 3.3 the difference between the respective error estimates and confidence intervals between the MLE approach and the worst-case minimax method. As expected from the results of [153], the optimal minimax method provides a more conservative bound on the error estimate. Furthermore, the resampling procedure is clearly highly sensitive to the initial set of outcomes, making it difficult to provide a reliable error bound via the MLE method. For this particular set of outcomes, the empirical confidence interval does not even contain the true fidelity.

We then generate an optimized experimental protocol, initialized with the above DFE-based protocol, and compare the results in Figure 3.4. In this example, we can see that the width of the minimax confidence interval has decreased, as expected. We also observe that the empirical MLE confidence interval is roughly the same length, but now overlaps the true fidelity.

However, the comparison between Figures 3.3 and 3.4 is questionable at best, as clearly the quality of the resampled data is far more important to the MLE fidelity estimation analysis than the allocation of the original measurements. This is a known shortcoming of the Monte Carlo approach, and in turn a limitation of the MLE method, as there is no associated rigorous risk estimate. For the purposes of demonstration, however, we know exactly the state σ from which we generate our initial set of outcomes. This means that we can generate additional sets of outcomes directly from the ground truth, and consider the MLE fidelity estimation procedure paired with a *ground-truth* empirical confidence interval.

We use the same 3-qubit adversarial state from Section 3.6.1, and generate $N = 5000$ sets of outcomes across various noise levels. We then generate fidelity estimates with both the MLE and minimax procedures, and present the distribution of each set of estimates in Figure 3.5, along with the minimax confidence interval, an empirical confidence interval of the minimax fidelity estimates, and an empirical confidence interval of the MLE fidelity estimates.

As seen previously in Figure 3.1, the minimax confidence interval is exactly the same across

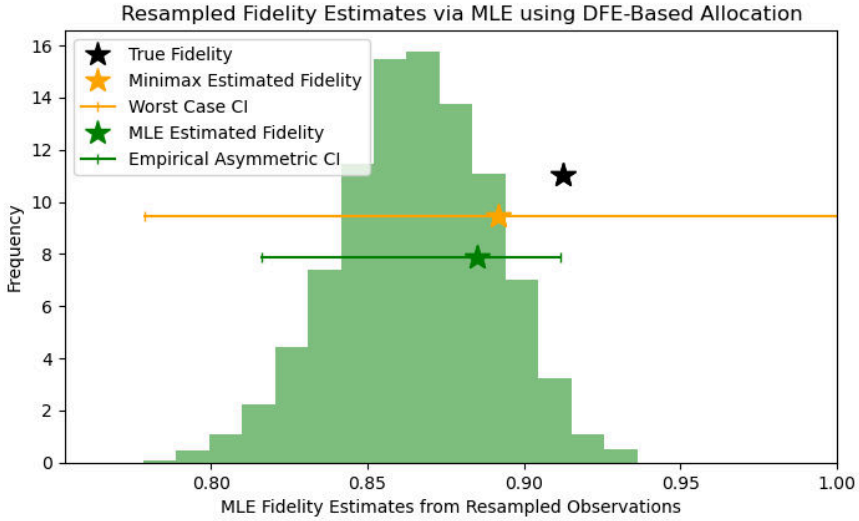


Figure 3.3: Comparison of minimax confidence interval to empirically resampled MLE interval for a DFE-based protocol. As expected, we see that the minimax method provides a more conservative bound on the error in the fidelity estimate, but the MLE confidence interval does not contain the true fidelity at all.

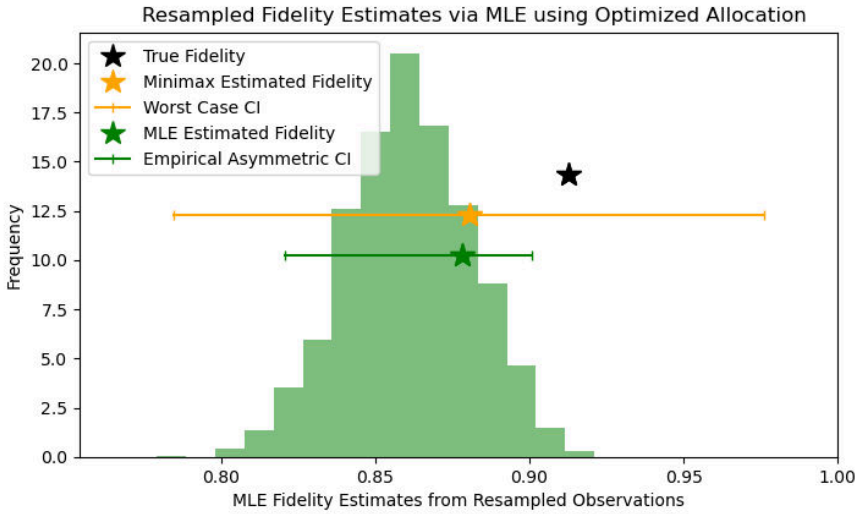


Figure 3.4: Comparison of minimax confidence interval to empirically resampled MLE interval for our optimized protocol. Compared to the conventional DFE-based protocol, the optimized protocol analyzed with MLE results in a marginally improved risk estimate. Note, however, that this demonstration is performed on only a single set of observables, and does not represent a necessary consequence of optimizing the experimental protocol.

noise levels, and consequently the distribution of fidelity estimates is similar across noise levels (aside from the 1% noise level case, in which the upper limit of $\widehat{F} = 1.0$ is achieved). In contrast,

the variance of the MLE fidelity estimate distribution decreases as the noise level does, as shown by the associated confidence interval. At the same time, the distribution of MLE fidelity estimates is consistently less accurate than the minimax alternative.

We also observe in this example that at each noise level, the width of the MLE confidence interval *decreases* as a result of the optimization procedure. Although Figure 3.5 demonstrates this for only a single set of ρ and σ, we note that these results persist across other randomly constructed target states and noise models for σ as well. This is a somewhat unexpected result, as the criterion with which our optimized experimental protocol is generated is entirely unrelated to the width of the MLE confidence interval, which itself is constructed without reference to the true target state. It further suggests that the measurement allocation discovered by our optimization procedure is based on some fundamental, yet obscure property of the underlying state.

3.7 Conclusion

The optimal minimax method is able to provide rigorous error bounds on the estimate of fidelity for a given measurement protocol. Because this method computes these error bounds in the absence of any experimental data, it permits numerical optimization over the different available measurement settings in such a way that experimental resources can be allocated more intelligently than by conventional techniques. We see that this approach not only reduces the calculated minimax risk in quantum fidelity estimation, but also leads directly to improvements in the fidelity estimates themselves.

There are many augmentations to the optimization framework which could be made so that the resulting experimental protocols have further utility in practical implementation. As one example, measurement sparsity is hugely important in an experimental context, and should often be prioritized even beyond the inherent sparsity obtained from using fine POVM measurement settings. This is simply because switching measurements is a particularly expensive and time consuming part of the verification process. With the understanding that even small changes in the measurement protocol can severely improve the resulting fidelity estimator, modifications to the

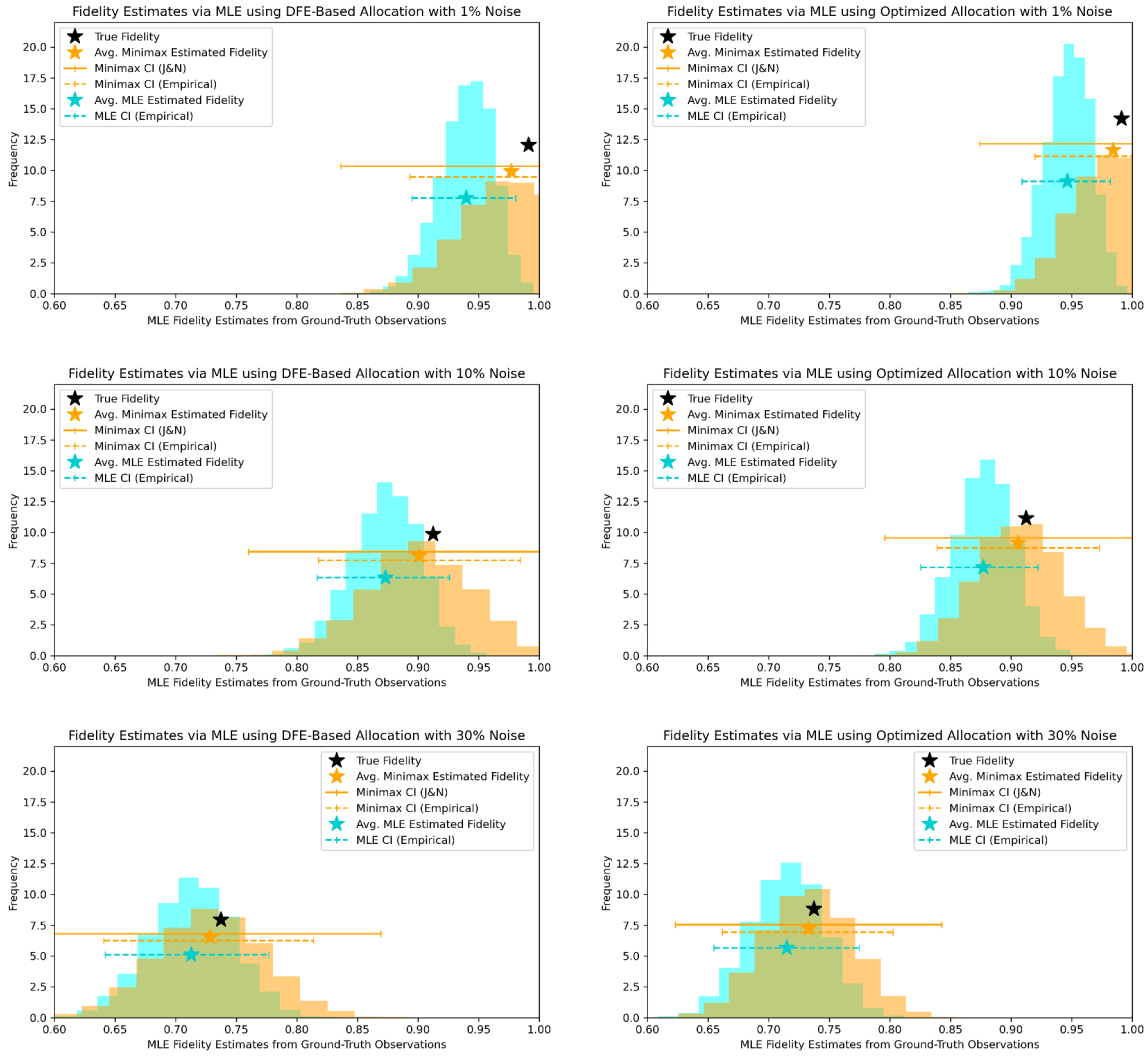


Figure 3.5: We compare distributions of fidelity estimates generated by the minimax procedure and an MLE procedure. We also plot the confidence interval provided by the minimax method around the average minimax fidelity estimate, and an optimal empirical confidence interval for each distribution .

minimization problem in Equation (3.10) to directly encourage sparsity in the vector of shot counts have the potential to greatly improve efficiency in verifying modern quantum devices.

Finally, the results of this chapter demonstrate the potential of methods which improve the results of *other* methods of statistical analysis, most notably MLE. In context of Equation (3.16), this is equivalent to selecting an optimal *weighting* of measurements w_i which most improves performance along some performance metric. This alternate problem context poses a number of

challenges. In contrast to the proposed work, in which the relevant method of analysis provides a straightforward criterion in the risk $\widehat{R}$, there is no directly equivalent metric for MLE, although metrics which measure uncertainty in the reconstruction [116] or the proximity between the target and experimental states [136] should suffice. Furthermore, such a method would require some measure of this criterion *prior* to the collection of experimental data. As is the case for the minimax method, this could be addressed through some form of worst-case error analysis, or perhaps consider Equation (3.16) which fundamentally includes uncertainty in the frequencies f_k for each POVM. In either case, MLE is an already common methodology for quantum state tomography, and so any technique that describes an associated experimental protocol could see similar widespread use.

Part II: Geometry Processing for Physics Simulations

Chapter 4

Introduction to Part II

Our primary motivation to pursue problems in computational geometry is multiphysics simulation, in which the interactions between geometrically complicated shapes are modeled on a computational domain through the interaction of governing PDEs. A practical challenge for such problems is their initialization, as the geometric shapes of interest are rarely defined in a way that is immediately compatible with the physics solver used. For example, in frameworks like the multi-material Arbitrary Lagrangian-Eulerian method, it is not the shapes themselves which are subject to the governing equations, but rather some kind of derived data [11, 121]. Most commonly, this data is taken to be the volume of the overlap between the shape and each computational cell in the simulation, but other type of geometric data can be useful as well, such as more general geometric moments [40].

We address this challenge through the two specific subproblems depicted in Figure 4.1: **Shaping**, and **Interface Reconstruction**.

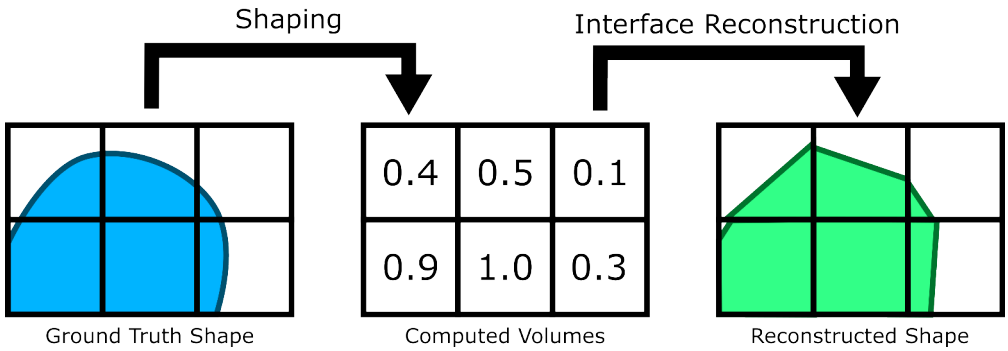


Figure 4.1: Material Interface Reconstruction as an inverse problem for Shaping

We refer to the process of collecting volume fraction data from arbitrary geometric input as *shaping* [180]. As a motivating example in 2D, we consider a material covering a domain $D \subset \mathbb{R}^2$, and a computational cell covering a domain $C \subset \mathbb{R}^2$. Naturally, the volume fraction V for material D within C is described by the integral

$$V = \frac{1}{|C|} \int_{D \cap C} dx. \tag{4.1}$$

While this can be solved directly for simple kinds of shapes, direct evaluation over intersecting regions is difficult even for complex numerical integration schemes [58]. A less accurate, but considerably more flexible approach is to instead sample the computational cell C with quadrature nodes, and evaluate

$$V = \frac{1}{|C|} \int_{C} I_D(x)\, dx, \tag{4.2}$$

where the indicator function $I_D(x)$ is typically easier to construct and evaluate for a general shape D.

However, the reliance on an indicator function makes these methods susceptible to a very common category of geometric errors. In cases where the underlying model is not *watertight*, in the sense that the numerical specification of the shape contains gaps and/or overlap between adjacent components, the precise definition of an indicator function is mathematically dubious at best, and computationally disastrous at worst.

It is in this context that we introduce the primary mathematical instrument of the first and second chapters of Part II, a *Generalized Winding Number* which is capable of providing a rigorous definition of "containment" for objects which otherwise have no volumetric interior. In practice, we simply substitute

$$V = \frac{1}{|C|} \int_{C} I_D(x)\, dx = \frac{1}{|C|} \int_{C} \text{round}\left(w_D(x)\right)\, dx, \tag{4.3}$$

and our shaping algorithm becomes robust to categories of input geometry which would otherwise be completely inapplicable. The principle contribution of these first two chapters in Part II is the definition of algorithms for the efficient and accurate calculation of the generalized iwnding

number for common types of explicitly defined geometric objects, respectively parameteric curves in 2D and trimmed NURBS surfaces in 3D. The content of Chapter 5 is largely based on the work **Robust Containment Queries over Collections of Rational Parametric Curves via Generalized Winding Numbers**, published in ACM Transactions on Graphics in July 2024 [163]. The content of Chapter 6 is based on the forthcoming manuscript **Robust Containment Queries over Collections of Trimmed NURBS Surfaces via Generalized Winding Numbers**, which can currently be found on arXiv [169].

We now turn our attention to the second half of Figure 4.1, the problem of *material interface reconstruction*. While many methods of multiphysics simulation can proceed entirely from specified moment data, there are many contexts in which it is necessary to explicitly reconstruct the geometric object whose data matches that provided by the simulation. Principal among these is visualization, which is essential for a thorough analysis of the results of such a simulation. Other times, the reconstruction method is essential for accurate simulation of the physics. For example, the *forward* motion of material can be calculated through the *backward* motion of each computational cell, computing geometric data for the overlap of the current reconstruction (potential via shaping) and the backtraced cell.

In many ways, the reconstruction problem is fundamentally underdetermined, as there are many distinct shapes which are defined by the same geometric data. As a result, nearly all reconstruction methods require restricting the space of representable objects to make the necessary optimization problem tractable, most commonly by requiring that the reconstructed interface have some particular form. For example, the cartoon in Figure 4.1 performs its reconstruction using linear elements. In any case, the reconstructed interface is nearly always an approximation of the ground-truth geometry.

Nevertheless, recent advances in the interface reconstruction problem are very concerned with increasing the accuracy of the reconstruction through an optimization strategy which can consider more complex interfaces as candidate reconstructions. However, methods in the literature which do so in 3D are, at the time of writing, predominantly limited by the requirement of considering

multiple computational cells simultaneously. This is the motivation for the third chapter in Part II, where we devise a strategy for the 3D reconstruction problem that is capable of generating more complex intersections through the consideration of multiple linear elements simultaneously. In contrast to contemporary approaches, our method is capable of creating these complex reconstructions using only the moment data from a single computational cell. In total, the content of Chapter 7 is based on the forthcoming work **Multi-Plane Moment-of-Fluid Interface Reconstruction in 3D** which can currently be found on ResearchGate [166].

As suggested by Figure 4.1, one can consider interface reconstruction to an inverse problem for shaping: one is concerned with generating aggregated data from the geometric objects occupying a computational cell, and the other requires generating some approximation of the geometric object from the collection of data. However, the dominant theme in our approach to each problem is that our methods are both *robust* and *reliable*, in the sense that they can operate on a wide variety of imperfect problem scenarios and fail gracefully and predictably in cases where an exact solution cannot be found.

Chapter 5

Generalized Winding Numbers for Rational Parametric Curves

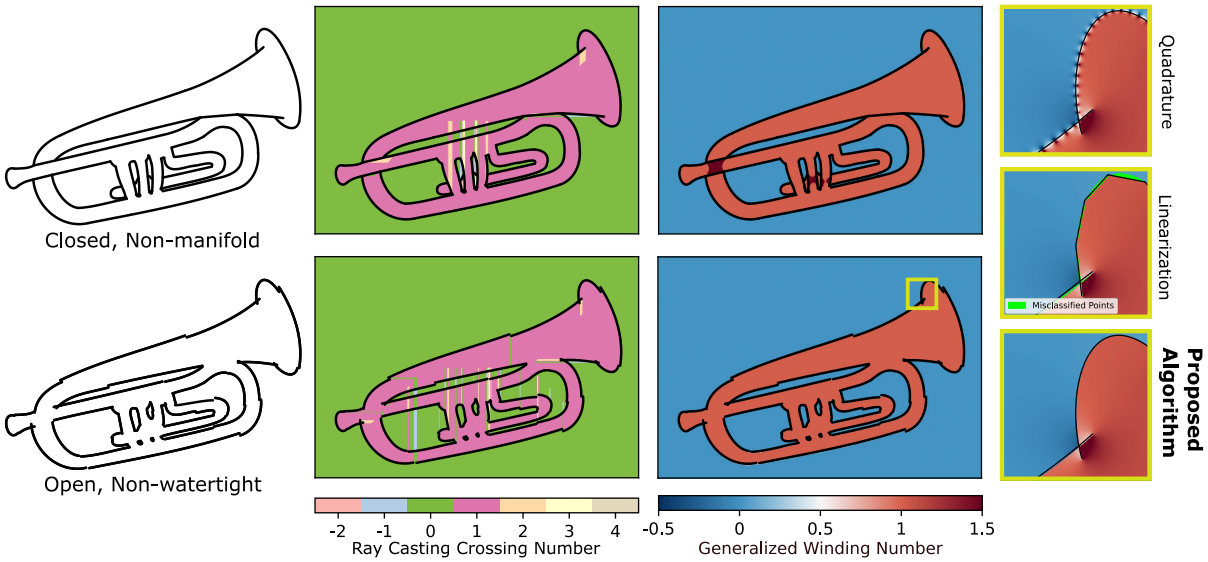


Figure 5.1: Evaluation of 2D point containment queries is often hindered by undesirable, but often unavoidable features of the bounding geometry such as non-manifold or non-watertight edges (left). This causes catastrophic errors for traditional approaches to deciding containment, in particular those that use ray casting (middle left). Because the field of generalized winding numbers degrades smoothly around geometric errors, they are well-suited for use in robust containment queries (middle right). In this chapter, we present a framework for evaluating exact generalized winding numbers for arbitrary collections of curved objects, as well as a novel point-in-curved-polygon algorithm that facilitates their efficient calculation (right).

In collaboration with David Gunderman and Kenneth Weiss

5.1 Abstract

Point containment queries for 2D regions bound by watertight geometric curves, i.e., closed and without self-intersections, can be evaluated straightforwardly with a number of well-studied algorithms. When this assumption on domain geometry is not met, such methods are either unusable, or prone to misclassifications that can lead to cascading errors in downstream applications. More robust point classification schemes based on generalized winding numbers have been proposed, as they are indifferent to these imperfections. However, existing algorithms are limited to point clouds and collections of linear elements. We extend this methodology to encompass more general curved shapes with an algorithm that evaluates the winding number scalar field over unstructured collections of rational parametric curves. In particular, we evaluate the winding number for each curve independently, making the derived containment query robust to how the curves are arranged. We ensure geometric fidelity in our queries by treating each curve as equivalent to an adaptively constructed polyline that provably has the same generalized winding number at the point of interest. Our algorithm is numerically stable for points that are arbitrarily close to the model, and explicitly treats points that are coincident with curves. We demonstrate the improvements in computational performance granted by this method over conventional techniques as well as the robustness induced by its application.

5.2 Introduction

In the fields of computer graphics, Computer Aided Geometric Design (CAGD) and Computer Aided Engineering (CAE), it is common for geometric objects to be expressed using a boundary representation (B-Rep). We consider these B-Reps to be defined via non-uniform rational B-spline (NURBS) curves in 2D. Such objects are invaluable in the design of CAD models, as they allow for precise geometric control of the object's boundary [132]. However, it can still be desirable in many contexts, such as in animation or simulation, to treat the interior volume of these objects explicitly [112].

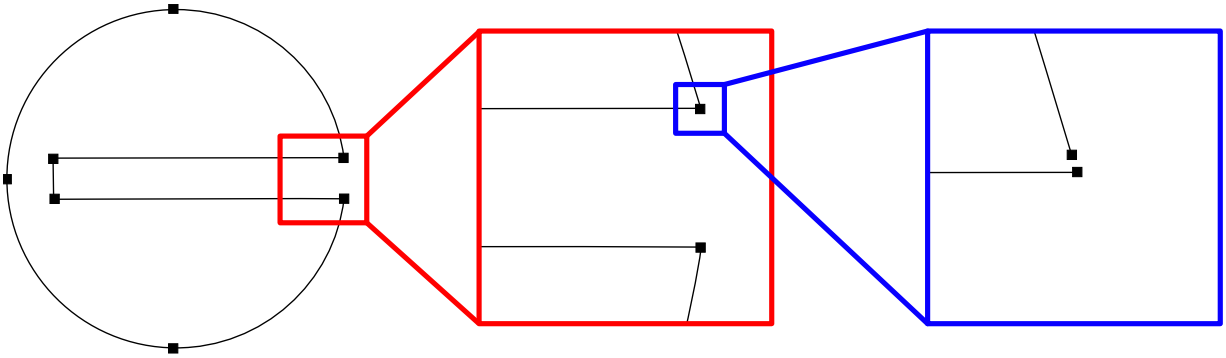


Figure 5.2: Geometric errors can be visually imperceptible, but can still cause a shape to have no topological interior.

Such a task motivates development of the *containment query*, a geometric predicate that returns whether an arbitrary spatial location is contained within an arbitrary shape. The rapid evaluation of containment queries is quite well understood for many classes of shapes, such as polygons and regions bounded by Bézier curves [120, 24, 129]. However, such methods assume the boundary is *watertight*, having no gaps between connected components. In reality, the vast majority of B-Reps are created by hand within some form of CAD software, leading to so-called "messy" CAD geometry.

A straightforward source of these problems is human error during design or construction, but more subtle issues can be introduced through varying tolerances within and between different software tools. This results in boundary models with human-imperceptible, but numerically significant gaps and overlaps between individual components of the boundary [111] (see Figure 5.2). Even in situations prioritizing visibility over strict geometric fidelity, such as in computer graphics and animation, care must be taken to ensure that these errors are properly accounted for. For example, the *dynamic planar map* heuristic used by Adobe Illustrator's Live Paint Bucket tool allows users to set fills for closed regions bounded by intersecting collections of Bézier curves and exposes a global threshold parameter to connect gaps between curves in the collection [5].

As an additional class of motivating examples, we consider applications in multiphysics simulation, for which containment classification errors can lead to cascading problems and incorrect

results. The mathematical underpinnings of containment within non-watertight shapes is dubious, which leads to a tenuous theoretical foundation for established methods that assume geometric continuity. This is problematic, as errors are frequently inconsistent within local regions, with even arbitrarily close points receiving different classifications. More practically, these errors can result in unexpected and fatal errors in simulation software, wasting computational resources and developer time [148].

At the same time, physical applications often prevent approximating curved geometry with piecewise linear segments. Although this would allow for treatment of the messy—but now linear—geometry with established methods, such an approximation can produce sizable errors in the relevant numerical method [154]. As a concrete example, multimaterial Arbitrary Lagrangian-Eulerian (ALE) methods require additional preprocessing to initialize the volume fraction of each material in a computational cell [69, 11]. This, in turn, necessitates classifying each point in a quadrature rule along the background grid as inside or outside the volume bounded by the corresponding shape [180]. It is critical that this identification takes into account the full curvature of the bounding geometry, as piecewise linear approximations can cause quadrature points near material boundaries to be misclassified, unpredictably compounding the error in the calculation.

Such geometric errors can be resolved with techniques in surface repair [113, 127, 16], which alter the B-Rep itself to an approximated surface that lacks these imperfections. However, doing so by hand can be impractical for the typical number of imperfections in such a model, and automated procedures during geometry pre-processing can inadvertently remove important model features. Instead, we require a containment query that is *robust* to such artifacts, in the sense that an appropriate inside/outside classification is always returned, independent of size and frequency of these boundary errors.

To address these concerns, we present in this chapter three principal contributions:

- We extend the theoretical framework of generalized winding numbers [79] to the context of an unstructured collection of 2D rational parametric curves, from which one can derive a

robust containment query that indicates whether an arbitrary point should be considered interior to non-watertight geometry.

- We address the issue of defining and evaluating generalized winding numbers for points that are coincident with the boundary. This discussion lies outside the scope of much of the existing literature, as such points require unique consideration in the context of curved geometry.

- We provide a novel algorithm for evaluating integer winding numbers with respect to curved geometry, which is the principal component of our technique for evaluating generalized winding numbers on such shapes. We demonstrate that this calculation is robust to non-watertight geometry and considerably more performant than state-of-the-art techniques.

Altogether, these contributions allow for robust containment queries of objects implied by messy 2D CAD geometry, composed of collections of parametric curves (see Figure 5.1). Our algorithm has been implemented in Axom, a BSD-licensed open-source library of Computer Science infrastructure for HPC applications [23].

5.3 Background and Related Work

Even in the ideal and watertight setting, methods of evaluating containment vary wildly depending on the numerical representation of the bounding geometry. As simple examples, the operation can be performed trivially for axis-aligned quadrilaterals through the evaluation of inequalities, for spheres by computing a distance to the center, and for triangles using Barycentric coordinates. More complicated procedures are necessary for point containment in arbitrary polygons [64]. These algorithms can generally be broken down into two categories (see Figure 5.3).

Ray Casting algorithms determine containment by extending a ray from the specified query point out to infinity, and counting the number of times the ray intersects the domain boundary to produce a *crossing number* [160]. As a point moves along this ray, it will alternate between interior and exterior, and therefore an odd (even) crossing number indicates the point is interior (exterior).

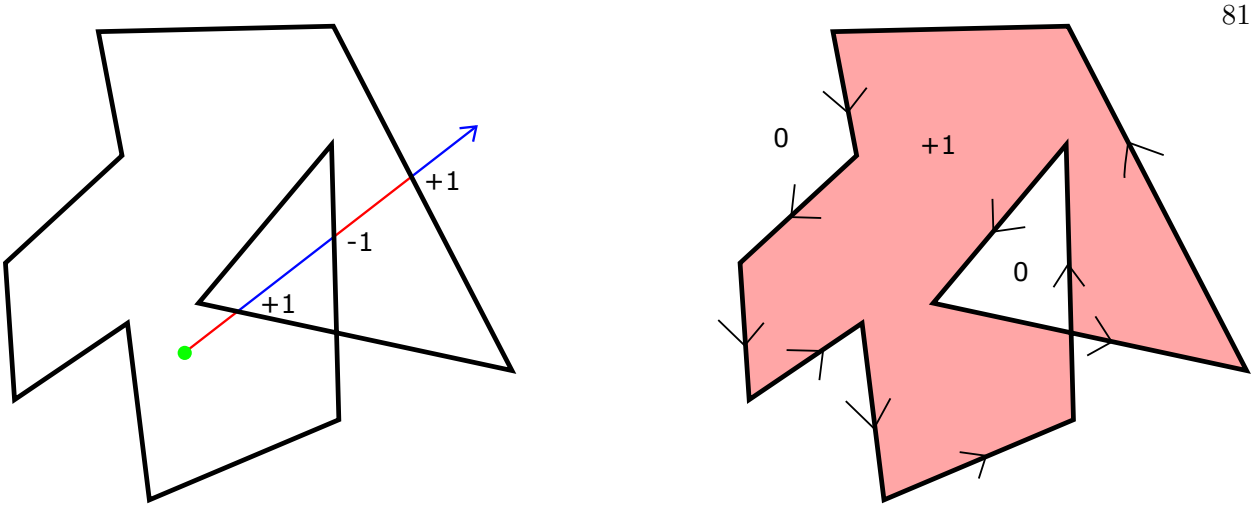


Figure 5.3: Given an arbitrary query point, containment can be determined with a ray casting algorithm that counts (signed) *intersections* between the polygon and a ray extending from the query (left), or a winding number algorithm that counts *revolutions* of the curve around the query (right).

Computing intersections between the given ray and the linear edges of a polygon can be done very efficiently, with many graphics processing units (GPUs) having dedicated ray-tracing cores that can perform the operation massively in parallel [185]. However, these methods are sensitive to the specific ray that is extended, needing to account for special cases when the ray intersects a polygon vertex or entire edge, which can be unnecessarily more costly [41, 162]. *Winding number* algorithms are a class of algorithms that are more indifferent to these issues, and, as such, are the type from which we derive our proposed method. In a winding number algorithm, we instead count the number of revolutions around the query point made by a particle traveling on the domain boundary [70]. This value can be considered a generalization of the crossing number, and can be used to determine containment with the same even-odd rule. By virtue of not being dependent on any particular extended ray, these algorithms implicitly handle edge-cases that are problematic for ray casting. This means even in the context of simple, watertight geometry, such algorithms can be considered a more robust approach to containment queries.

Containment of arbitrary points in more general curved shapes is a much more difficult problem, and has been studied extensively in the context of 2D vector graphics. As an example, we refer

to the process of *linearization*, in which a curve is approximated by connected linear segments, an approach that is fairly common in both modern graphics engines and physics codes. In the former, information about the targeted display can be leveraged to ensure that the approximating polyline is constructed with sufficiently high resolution such that it is indistinguishable to a viewer [88]. Combined with the fact that the polyline must be rendered with some predetermined thickness, it is therefore theoretically possible (although rarely enforced) for containment queries on linearized watertight geometries on a rendered display to have no *observable* misclassifications.

On the other hand, this type of polyline approximation of curved geometry is known to cause severe and often unexpected errors in the modeling of physical systems through PDEs and finite element analysis [154]. This is a particular problem in the case where points of interest are allowed to be placed arbitrarily close to the curve, as then no guarantee can be made as to how many refinements in the approximation are necessary to achieve perfect geometric fidelity. In place of such a linearization, the use of NURBS to represent boundaries in a finite element analysis has been quite successful in reducing or even eliminating geometric error [154, 155]. As a result of this, we consider operating directly on curves without any approximation to be a necessary component of our proposed algorithm so as to preserve the accuracy of an underlying numerical method.

We similarly desire an algorithm that can reach this level of geometric fidelity for a very general class of curves. For example, cubic Bézier curves are the highest order required by postscript and SVG formats, as the level of detail they afford is typically sufficient for design applications. However, applications in computational multiphysics can necessitate an algorithm that works for *rational* Bézier curves of *arbitrary* order. This is because the NURBS shapes that are used by more sophisticated modeling software can always be decomposed into a collection of (in principle arbitrary order) rational Bézier primitives through the process of Bézier extraction [132, 173]. However as we will see, the proposed algorithm is ultimately indifferent to the order and/or rationality of the curve in question.

Containment within regions defined by Bézier curves has also been studied in the context of 2D parametric trimming curves for 3D NURBS surface patches. In this sub-problem, it must

be determined if a point in the 2D parameter space of the 3D surface is contained within the (often, but not necessarily watertight) trimming curves. Because this and other contemporary 2D containment problems are frequently solved with ray casting algorithms [111], we note that they can largely be divided into two categories. In the algebraic approach, intersections between rays and arbitrary curves can be computed using typical root-finding techniques in parameter space [129], although standard bisection and gradient descent methods can only identify at most a single point of intersection. In contrast, geometric approaches take place in physical space, decomposing the curve into subcurves until intersections with the ray can be identified [46]. The current state-of-the art method, *Bézier clipping* [150, 120], lies at the intersection of these two approaches, where recursive algebraic subdivision is accelerated using the convex hull property of a Bézier curve. However, such methods are known to possibly report incorrect intersections and suffer from inefficiency in the presence of multiple spatially equivalent intersection points [42].

These issues are worsened by geometric errors present in the shape. As an example, if the arbitrary ray is by chance extended through a very small gap in the model, then the point must be classified as "exterior" despite this being unintuitive and contrary to the intentions of the mesh designer. Non-manifold edges in a shape are similarly problematic for ray casting methods. These issues can be somewhat mitigated by extending several rays and taking a consensus, but such approaches impose an additional computational burden while still producing noisy and potentially inconsistent classifications for nearby points [123]. We now show that, in addition to being indifferent to the specific edge cases introduced by ray casting, winding number algorithms can be extended such that they are also robust to more general issues introduced by non-watertight, non-manifold, curved geometry.

5.4 Generalized Winding Numbers

The generalized winding number is an extension of the standard integer-valued winding number to (potentially) non-watertight regions. While integer winding numbers partition the enclosed regions of a domain, the generalized winding number generates a harmonic scalar field that smoothly

degrades in the presence of discontinuities and self-intersections. This allows for the calculation of robust containment queries in the presence of messy geometry. As an example, a simple rule for handling the fractional values that result from imperfections in the bounding geometry is to round them to their nearest integer.

Such generalizations of winding numbers have recently shown great utility in geometry processing applications. In the context of both (linear) triangle meshes [79] and point clouds [10], the resulting inside/outside classification can be used to generate volumetric triangle/tetrahedral meshes out of completely unstructured geometric data. Additionally, the smooth degradation of the winding number field naturally introduces desirable locality properties, which in turn lead to more robust and performant algorithms in, for example, exact mesh booleans [175]. More recently, these particular winding number algorithms have been used in the context of immersogeometric analysis to simulate fluid flow around geometric objects defined by unstructured point clouds [8]. It is speculated in [58] that one could extend the calculation of generalized winding numbers by utilizing their own novel quadrature schemes for such objects [59], but preliminary tests of this approach are limited by the accuracy of numerical integration. Despite this recent surge in use, methods that compute *exact* generalized winding numbers for more general curved geometric objects are, to our knowledge, currently absent from the literature.

Nevertheless, the problem of computing generalized winding numbers for curved shapes has close analogues in the fields of both computer graphics and boundary integral equations. In the former, a connection can be drawn to diffusion curves, a primitive in vector graphics that also produces a scalar field defined by the solution of a differential equation with respect to boundary curves (albeit with different data prescribed on the curve) [79]. These applications have the advantage of being solved on a grid generated by a rasterized domain, meaning that a global solution can be obtained efficiently using a geometric multi-grid method [125]. [144] recently introduced a grid-free approach in which random walks and projections onto curved geometry are used to solve certain classes of PDEs, including that which implicitly defines the generalized winding number, focusing on improving speed and locality at the cost of some degree of accuracy and consistency. Within the

context of boundary integral equations, calculating winding numbers is a special case of the more general problem of integrating double layer potentials with a constant density function. However, such algorithms are typically applied to closed domains with simpler (although still curved) types of geometry, precluding general use on arbitrary, messy CAD models [90]. Furthermore, our proposed algorithm relies exclusively on geometric principles, eliminating the need to use costly and potentially inaccurate quadrature schemes.

We begin by reviewing the formal description of a generalized winding number in $\mathbb{R}^2$. Given an oriented curve $\Gamma \subset \mathbb{R}^2$ and query point $q \in \mathbb{R}^2 \setminus \Gamma$, the winding number w describes the (potentially incomplete) number of times the curve travels counterclockwise around the point. In the case where Γ is a closed curve, the scalar field $w_\Gamma(q)$ is integer-valued, and induces a partition of $\mathbb{R}^2$ that can be interpreted as containment in the region bounded by Γ. In the following discussion, it is notationally convenient to assume that the query point is located at the origin, and that the curve Γ has been appropriately translated by q. The scalar field is invariant to such global translations, and so the winding number of Γ at q, $w_\Gamma(q)$, is identical to the winding number at the origin after translating the curve, $w_{\Gamma-q}(0)$.

When the curve is piecewise linear with $\Gamma = \cup_i L_i$, the winding number at q can be computed as the sum of *signed* angles θ_i subtended by each linear component L_i, given by

$$w_{\Gamma-q}(0) := \frac{1}{2\pi} \sum_i \theta_i. \tag{5.1}$$

Furthermore, the winding number of each individual component of the piecewise linear shape (and of straight lines in general) is given by

$$w_{L_i-q}(0) = \frac{1}{2\pi}\theta_i, \tag{5.2}$$

again where θ_i is the angle subtended by L_i at q (see Figure 5.4).

We can consider the case for a more general collection of arbitrary curves analogously. Let $\Gamma = \cup_i \Gamma_i$ be a collection of rational Bézier curves that constitute the (likely not watertight) boundary of a 2D CAD model. In this case, the winding number of a point q with respect to a single curve Γ_i

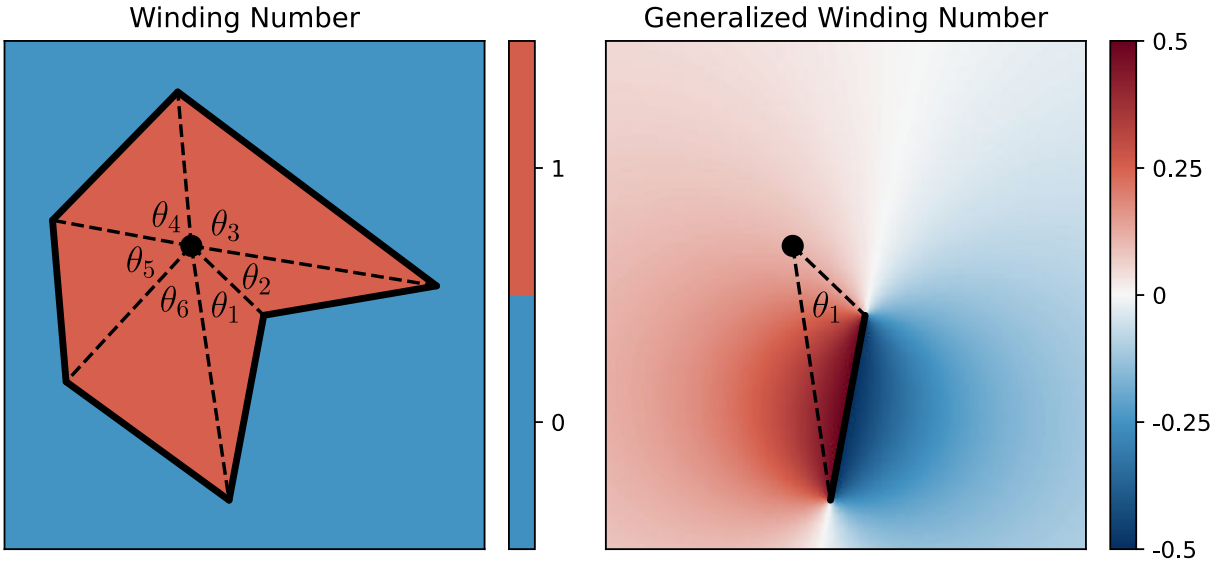


Figure 5.4: (left) The winding number for a piecewise linear shape can be computed by summing angles subtended by each segment. (right) Winding numbers for linear segments can be computed at arbitrary points independently of the remaining shape.

is given by

$$w_{\Gamma-q}(0) := \frac{1}{2\pi}\int_{\Gamma-q} d\theta = \frac{1}{2\pi}\sum_i \int_{\Gamma_i-q} d\theta \tag{5.3}$$

by the properties of the integration. This is notable, as it describes how the winding number for each individual component Γ_i can be computed completely independently of every other curve, with

$$w_{\Gamma_i-q}(0) := \frac{1}{2\pi}\int_{\Gamma_i-q} d\theta. \tag{5.4}$$

It is from this property of the generalized winding number that we derive both the robustness of our winding number calculation, and the robustness of the derived containment query. While the *collection* of Bézier curves may be messy in the sense that pairs of endpoints may not perfectly align, generalized winding numbers on each *individual* curve are well-defined and can be computed stably. At the same time, the sum of these values at a given point is exactly the generalized winding number of the entire shape, which can be used to produce intuitive and robust containment queries for non-watertight shapes (see Figure 5.5). Most commonly, this is done by rounding this value

to the nearest integer, and applying to it a conventional even-odd or non-zero rule to make the classification. Importantly, each of these conventions treat points with a zero winding number as exterior.

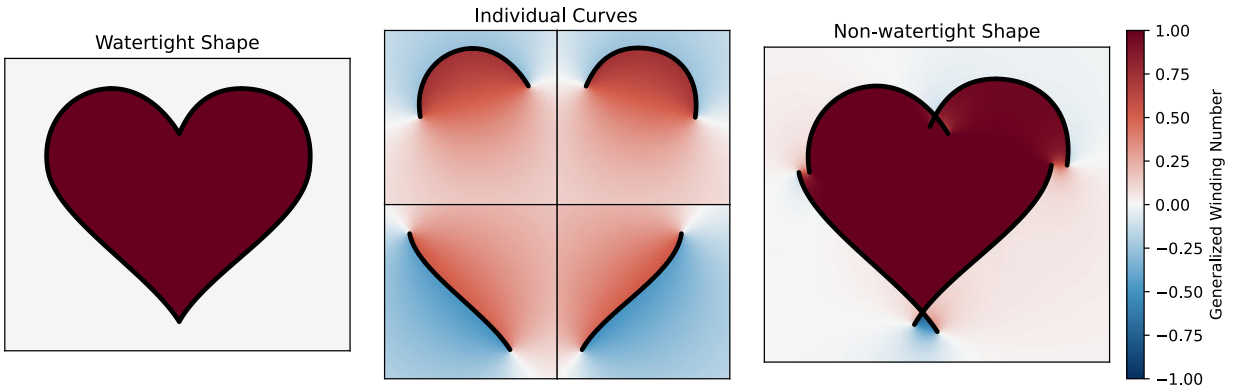


Figure 5.5: The GWN with respect to closed shapes (left) is always an integer. However, by independently summing contributions from each curved component (center), the GWN can also be computed over collections of curves containing gaps and overlaps (right).

Framed in this way, the remaining task is to evaluate generalized winding numbers for a single, arbitrary rational Bézier curve. As an initial approach, one could consider direct evaluation of the integral through numerical quadrature. When the curve is given parametrically as $(x(t), y(t)) \in \mathbb{R}^2$, $t \in [0, 1]$, as is the case for (rational) Bézier curves, we can evaluate this formula more practically in terms of Cartesian coordinates, as

$$w_\Gamma(q) := \frac{1}{2\pi} \int_0^1 \frac{x(t)y'(t) - x'(t)y(t)}{x^2(t) + y^2(t)}\, dt. \tag{5.5}$$

Direct evaluation of this integral through standard techniques is notoriously difficult. In particular, when Γ is close to the query point (or rather the origin under this translated notation) then the near-singular behavior of the integrand causes Gaussian quadrature to become unstable, as seen in Figure 5.6. Although some error is to be expected for this type of numerical integration, this error rapidly becomes unacceptable as the query point approaches the curve. Perhaps more concerningly, there is no immediate way to verify that the value returned by the method is accurate.

We can see how this would affect a containment query on a closed shape in Figure 5.6. In this example, the only correct values for the rounded winding number are 0, 1, and 2. However, even the most accurate quadrature scheme shown produces values in the vicinity of the quadrature nodes that are completely meaningless.

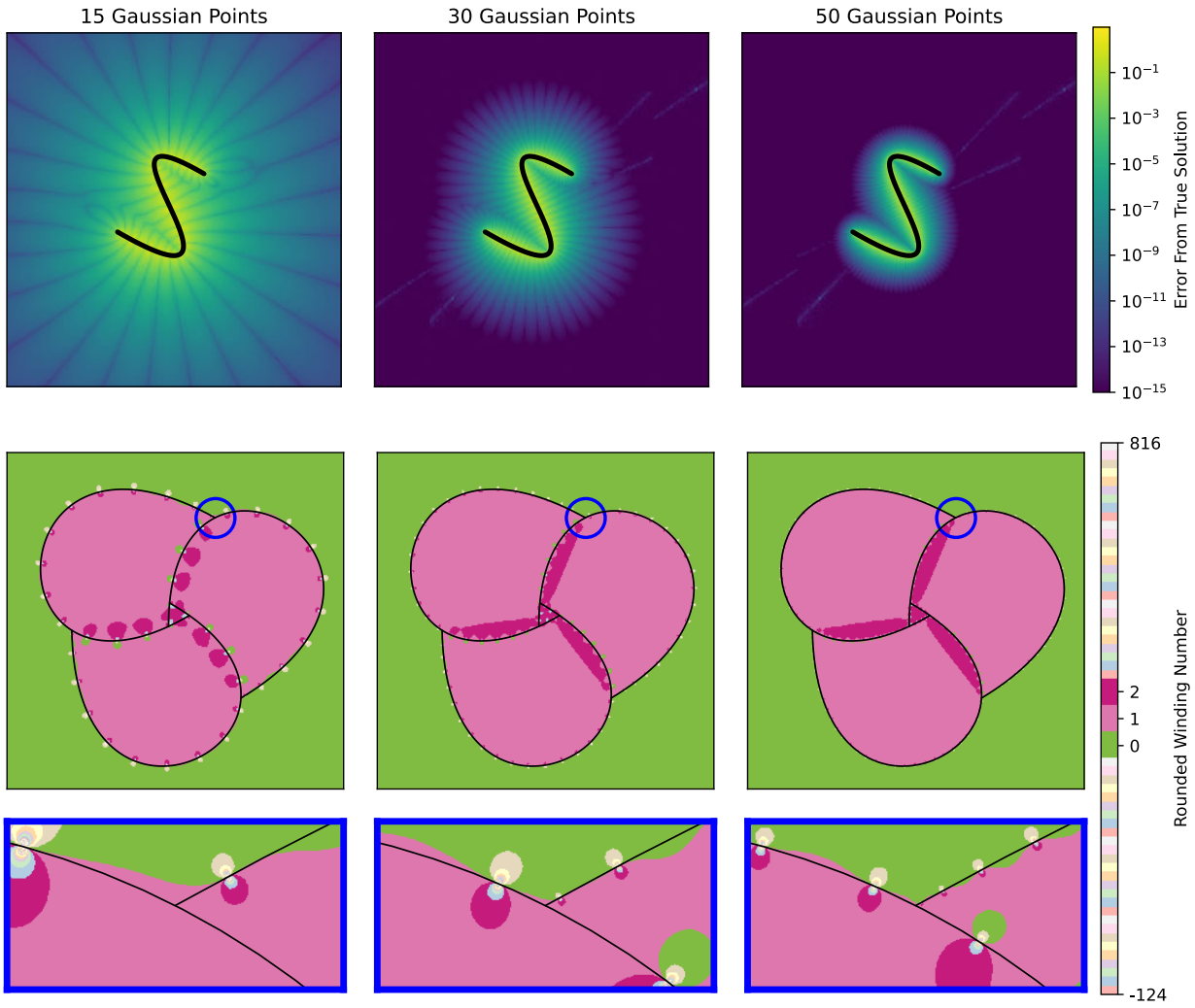


Figure 5.6: (top) The absolute error (log scale) in Gaussian quadrature used to compute the generalized winding number on a cubic Bézier curve with Equation 5.5, evaluated with 15-, 30- and 50-nodes. (middle) Using Guassian quadrature to compute generalized winding numbers over a shape leads to unacceptable errors. (bottom) Close-up of highlighted region.

5.5 Generalized Winding Numbers for Curved Geometry

While these integration formulae are useful from a theoretical perspective, their inherent instability necessitates an approach that avoids evaluating them directly. In their place, we develop a framework that computes winding numbers based only on geometric properties of the curve. To this end, we make heavy use of a particular kind of closing curve for each shape. Given an (in principle arbitrary) parametric curve Γ, we define the linear segment $C : [0, 1] \to \mathbb{R}^2$ by $C(t) = \Gamma(1)(1 - t) + \Gamma(0)t$ as the *linear closure* of our curve.

As the name suggests, the union of this closure C and Γ will always be a properly oriented, closed curve, and thus partitions $\mathbb{R}^2$ into discrete enclosing regions. Thus, winding numbers with respect to the total curve $\Gamma \cup C$ are integers, such that

$$w_\Gamma(a) + w_C(a) = w_{\Gamma \cup C}(a) \in \mathbb{Z}. \tag{5.6}$$

Most importantly, the winding number of such a closure can always be computed *exactly* without the need to appeal to quadrature by Equation 5.4, as the angle subtended by the introduced linear segment can be evaluated through a (relatively) inexpensive arccosine. In this way, we can always solve the problem of the *generalized* winding number using a solution to the *integer* winding number problem on the closed curve, as subtracting away the contribution of this linear closure can be done independently of the original curve geometry. While we present our own algorithm to compute this integer winding number in Section 5.6 that outperforms known alternatives, we note that this strategy is fully compatible with more conventional techniques for computing containment queries in closed regions bounded by curves, e.g., Bézier clipping [120].

This principle is particularly useful in the case when $w_{\Gamma \cup C}(q) = 0$, i.e., the query point is located outside the closed shape $\Gamma \cup C$, or outside the convex hull of Γ itself, where we have $w_\Gamma(q) = -w_C(q)$. This usage is introduced in [79], where it is used in a hierarchical evaluation of the winding number for collections of linear triangular facets. An important consequence of this for curved geometry is that for query points that are far enough away, we can reverse the orientation of C and treat the shape as *equivalent* to its straight line closure, and compute the winding number

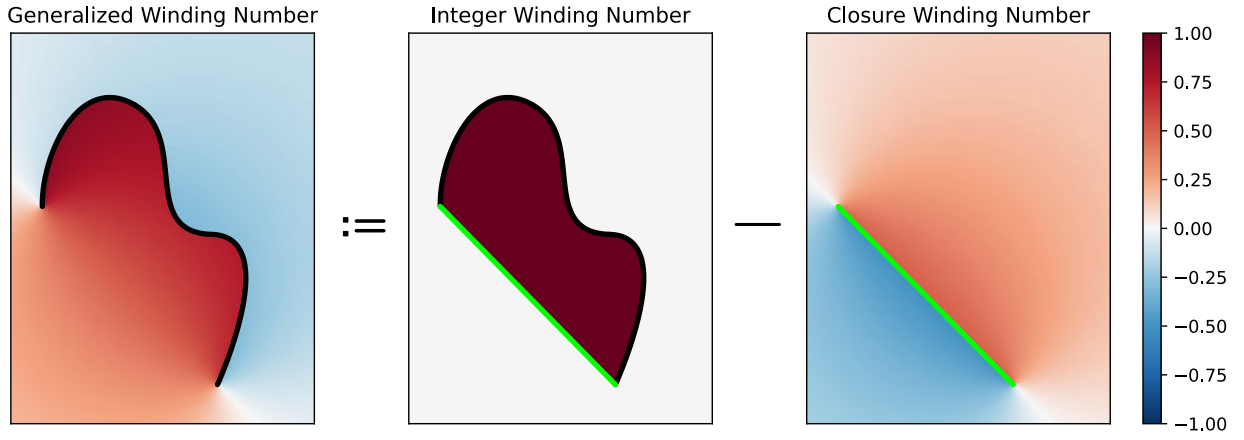


Figure 5.7: Overview of 2D GWN algorithm. The unknown winding number with respect to a curved shape (left) can be computed by finding the *integer* winding number of the closed figure (center) and subtracting away the contribution of its closure (right).

of the entire curved segment *immediately* and *exactly*.

We further extend this principle to evaluate generalized winding numbers for points that are arbitrarily close to the curve with the same assurances of exact accuracy. In such cases, Γ can be replaced with a piecewise linear approximation $\widetilde{\Gamma}$ so long as the *integer* winding number at q remains unchanged between $\Gamma \cup C$ and $\widetilde{\Gamma} \cup C$ (see Figure 5.8). Doing so necessarily leaves the *generalized* winding number at q unchanged as well, as we have

$$w_\Gamma(q) = w_{\Gamma \cup C}(q) - w_C(q) = w_{\widetilde{\Gamma} \cup C}(q) - w_C(q) = w_{\widetilde{\Gamma}}(q).$$

It is from these observations that we derive an algorithm for computing *exact* generalized winding numbers over a collection of curves. For both far and near query points, we construct appropriate linearizations of our curves that are guaranteed to have the same generalized winding number at the point of interest as their curved counterparts. For a given point, the vast majority of curves in the model will be considered far, and their contribution to the generalized winding number can be computed with a single arccosine evaluation. For the few curves that are close enough such that the linear closure itself is an insufficient approximation, we construct an approximating polyline that provably generates the same winding number. In such cases, we evaluate the integer winding number for the closed polygon, which can be done without reference to any trigonometric

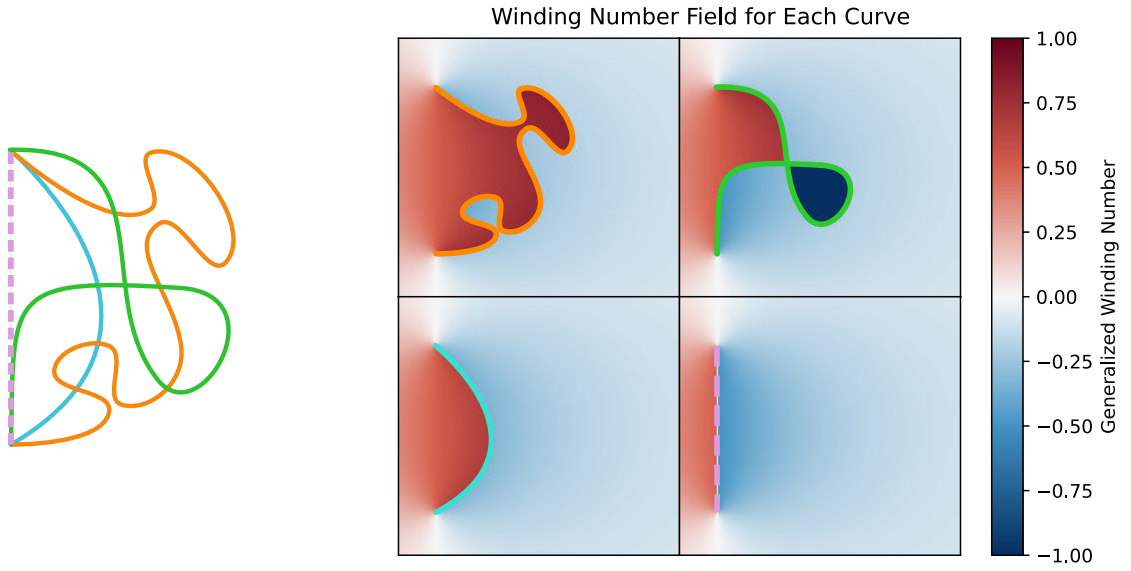


Figure 5.8: (left) All three curves are closed by the same dashed line, and the shaded region is exterior to all three closed shapes. (right) This means that the winding number field generated by each curve (and the closure, up to orientation) is *identical* in the shaded region.

functions using the point-in-polygon algorithm in [70] and subtract away the contribution of the linear closure.

In either case, we are able to compute the *exact* generalized winding number for an arbitrary point using only a single evaluation of arccosine for each curve in the shape. All that remains is to ensure that this polyline is adaptively constructed such that evaluating the integer winding number of the polyline is efficient.

5.6 Winding Number Algorithms

We now describe our complete algorithm for solving the generalized winding number problem for a rational Bézier curve at a given query point (Algorithm 3). The core of this algorithm is described in Algorithm 4, which computes the *integer* winding number for a closed rational Bézier curve without reference to ray casting.

In brief, given such a curve Γ, we adaptively construct a polyline approximate $\widetilde{\Gamma}$ which has the same winding number at the point of interest. We then apply a standard point-in-polygon

algorithm to this closed polyline to compute their shared integer winding number and subtract the contribution of their shared closure.

For this procedure to work, we require that our polyline $\widetilde{\Gamma}$ satisfies $w_{\Gamma \cup C}(q) = w_{\widetilde{\Gamma} \cup C}(q)$. The simplest way to ensure this is for q to be located outside both closed shapes $\Gamma \cup C$ and $\widetilde{\Gamma} \cup C$, as then this shared integer winding number is equal to 0. Because this cannot be guaranteed in general, we recursively bisect the curve into components $\{\Gamma_i\}_i$ until q is located outside the closed shape for each.

There are a number of ways to ensure that for each component, $w_{\Gamma_i \cup C_i}(q) = 0$. For example the convex hull property of a Bézier curve states that the curve Γ_i is completely contained within the convex hull of its control nodes. Given a description of this convex hull, classic point-in-polygon algorithms such as [70] can be applied to test for containment. This leads to the following outline for our algorithm:

- If the query point is located outside this convex hull, then it is guaranteed to be outside the closed shape $\Gamma_i \cup C_i$ and we replace the component Γ_i with the reversal of its linear closure C_i, as they now provably have the same generalized winding number.
- Otherwise, we bisect the Bézier curve and repeat the algorithm on each half. As a base case, we check whether the curve is approximately linear (Algorithm 5) as such segments can be added directly to the polyline.

Once this approximating polyline is fully constructed, we close it and use the same point-in-polygon algorithm, `PolygonWindingNumber`, to compute the integer winding number for the closed polygon. The contribution of this closure is then subtracted, and we are left with the exact generalized winding number for the curve. Figure 5.9 illustrates this algorithm for three successively closer query points to a Bézier curve.

To improve computational efficiency in the case where subdivision is expensive and exact geometric accuracy is not needed, we optionally provide in Algorithm 5 an early stopping criterion. We choose this particular heuristic because it is easier to evaluate than the distance between the

query point and the curve, and has a more intuitive spatial interpretation than a simple bound on the maximum number of curve subdivisions. In practice, Bézier curves are smooth nearly everywhere on their interior, and so the recursive bisection step needs to be done relatively few times before the query point is found to be outside the convex hulls of each component, even for points very close to the curve. Importantly, this causes the accuracy of Algorithm 3 and runtime of the procedure to both be reasonably insensitive to this numerical tolerance, as the algorithm is likely to terminate for these smooth curves well before there has been sufficient refinement for subdivisions to be considered linear. Indeed, in the numerical examples we provide, we set this value to machine epsilon and find no significant loss in performance (See Section 5.8.2).

We note that it can be problematic if the query point is located directly on the curve, as in such cases the winding number is not formally defined. Since we need our algorithm to be compatible to such input points, we discuss this case extensively in Section 5.7.

Despite the simplicity of the above procedure, directly computing containment within a Bézier curve's convex hull can be prohibitively expensive. Instead, we first determine if the query is contained in an axis-aligned bounding box that encompasses the convex hull. Although this can have a much larger area, the bounding box containment query is inexpensive and most points will be exterior to it. If it is not, we use a test from [103] to check if the control polygon (defined by a Bézier curve's control nodes) is already simple and convex (see Algorithm 6). If so, we perform the point-in-polygon test to determine if the query lies outside the convex hull of the control polygon. For efficiency, our implementation notes that when a curve is simple and convex, its subcomponent curves will be as well.

Altogether, the proposed algorithm is given by Algorithm 3. We note that this procedure is not meaningfully restricted to rational Bézier curves. Their focus throughout this work reflects their simplicity during calculation through manipulation of their control nodes, which itself justifies their ubiquity in application. More generally, the algorithm is applicable to any curve for which it is possible to construct a bounding box and a linear closure. In particular, extension to collections of NURBS curves is straightforward.

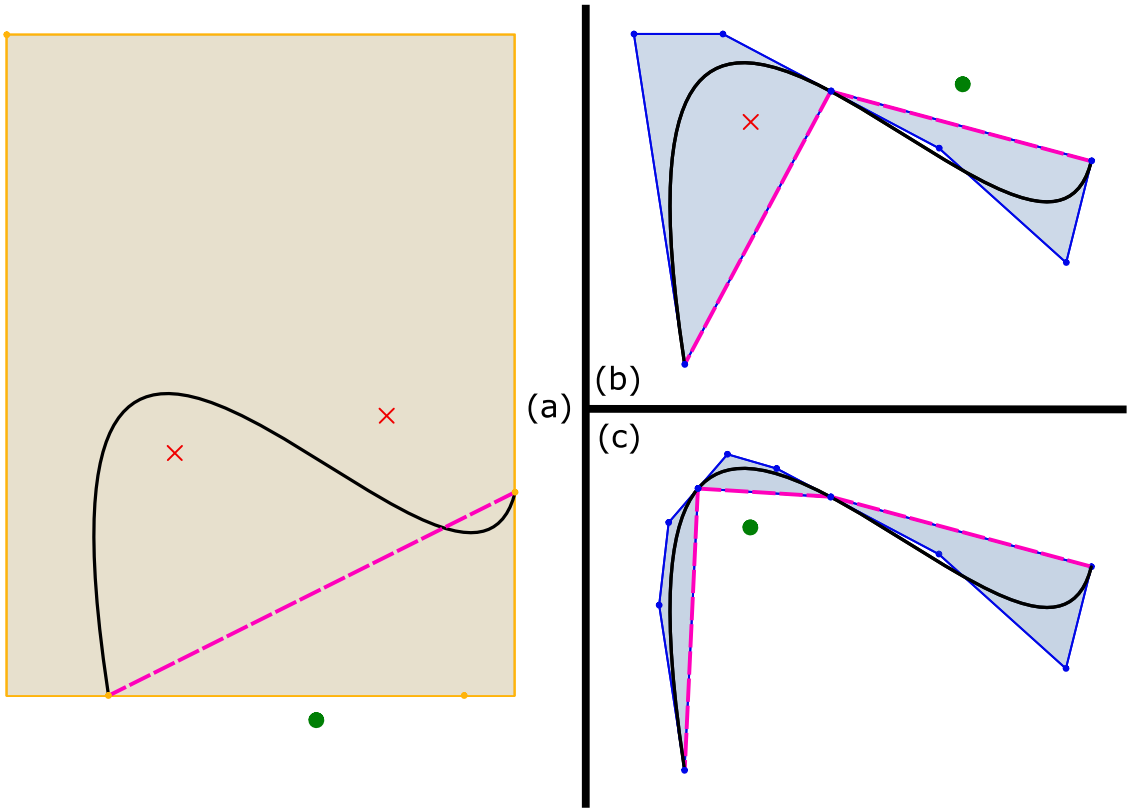


Figure 5.9: Three iterations of the approximating polyline algorithm. In (a), one point is outside the bounding box, and its winding number can be computed from the dashed closure. In (b), after a bisection we can compute the winding number for an additional point. We repeat the process in (c) to compute the winding number for the remaining point.

Algorithm 3: `WindingNumberCurve`: Evaluate the generalized winding number for an arbitrary rational parametric Bézier curve

---

Input: Γ: Rational parametric Bézier curve Γ
q: Query point
Output: w_Γ: The winding number evaluated at q

`/* Store the linear closure of` Γ `*/`
$C(t) \leftarrow (1-t)\Gamma(1) + t\Gamma(0)$
$w_C \leftarrow (1/2\pi) \times \left(\text{Signed angle subtended by } \overrightarrow{qC(0)} \text{ and } \overrightarrow{qC(1)}\right)$
if $q \notin$ `BoundingBox`(Γ) **then**
  return $-w_C$
else
  return `IntegerWindingNumberCurve`$(\Gamma \cup C, q) - w_C(q)$

Algorithm 4: `IntegerWindingNumberCurve` Evaluate the integer winding number for a rational parametric Bézier curve closed by a linear segment. We use an algorithm from [70] for `PolygonWindingNumber`.

Input: $\Gamma \cup C$: Closed rational parametric Bézier curve
q: Query point
Output: $w_{\Gamma \cup C}$The integer winding number evaluated at q

$\widetilde{\Gamma} \leftarrow \{\}$ `// Initialize the polyline approximation`
$w_{\Gamma \cup C} = 0$ `// Initialize the winding number`

Push Γ onto an empty stack.
while the stack is not empty do
  $\Gamma_0 \leftarrow$ `StackPop`
  $C_0(t) \leftarrow (1-t)\Gamma_0(0) + t\Gamma_0(1)$

  `/* Check for coincidence at the endpoints */`
  if `isSimpleConvex`(Γ_0) **and** $(q = \Gamma_0(0)$ **or** $q = \Gamma_0(1))$ **then**
    `/* Track the contribution of coincident points */`
    $w_{\Gamma \cup C}$ += `ConvexEndpointWindingNumber`(Γ_0, q)
    $\widetilde{\Gamma} \leftarrow \widetilde{\Gamma} \cup \{C_0\}$ `// Add to the polyline`
  else
    if $\{$`isSimpleConvex`(Γ_0) **and** $q \notin$ `ControlPolygon`$(\Gamma_0)\}$
    or `isApproximatelyLinear`(C_0)
    then
      $\widetilde{\Gamma} \leftarrow \widetilde{\Gamma} \cup \{C_0\}$ `// Add to the polyline`
    else
      $\Gamma_1, \Gamma_2 \leftarrow$ `Bisection`(Γ_0)
      `StackPush`(Γ_1, Γ_2)

$P \leftarrow \widetilde{\Gamma} \cup \{C\}$ `// Close the polyline, forming a polygon`
return `PolygonWindingNumber`$(P, q) + w_{\Gamma \cup C}$

Algorithm 5: `isApproximatelyLinear`: Return `true` if each control point of a rational parametric Bézier curve is within a given tolerance of the closure.

Input: Γ: Rational parametric Bézier curve
ϵ: User tolerance

$C_0(t) \leftarrow (1-t)\Gamma_0(0) + t\Gamma_1(1)$
for each control node P_i **of** Γ **do**
  if `SquaredDistance`$(C_0, P_i) \geq \epsilon$ **then**
    return `false`
return `true`

Algorithm 6: `isSimpleConvex`: Return `true` if the polygon of control points of a rational parametric Bézier curve is simple and convex (Adapted from [103])

Input: Γ: Rational parametric Bézier curve with control nodes $P_0, \cdots, P_n$

```
1 for i = 1, ..., (n − 1) do
      /* Store the linear segment connecting the nodes */
2     S(t) ← (1 − t)P_{i−1} + tP_{i+1}
3     if i ≤ n/2 then
4         if P_i and P_n are on the same side of S then
5             return false
6     else
7         if P_i and P_0 are on the same side of S then
8             return false
9 return true
```

5.7 Generalized Winding Numbers for Coincident Points

The generalized winding number is, in a strict mathematical sense, undefined for points located directly on the curve. This poses a practical problem during the implementation of a containment query, as they are often executed massively in parallel for large clusters of points without any *a priori* knowledge of their position relative to the boundary. Furthermore, our desire for exact geometric fidelity precludes us from applying the standard trick of perturbing such points randomly to place them definitively on one side of the singular boundary. In spite of this, it is necessary for compatibility with respect to downstream applications that the algorithm definitively return a value that adheres to a convention that is mathematically justified and intuitive to the caller of `WindingNumberCurve`.

An important aspect of the winding number scalar field is that it is harmonic, being the unique solution of a Laplacian operator with boundary data prescribed by each side of our curves, enforcing a jump discontinuity across them. Therefore, the simplest convention for the winding number of coincident points is to take it as the average value across this jump discontinuity. For linear segments, this boundary data enforces that the winding number approaches $+1/2$ from one side and $-1/2$ on the other. This means that the winding number should be exactly 0 at every

coincident point along linear segments. Similarly, this convention would ensure that the winding number along a more general closed shape is a fixed half-integer value along its length, changing only across self-intersections. However, this presents a unique problem for open, curved shapes, as the winding number is no longer constant along the length of the curve. This underscores the importance of a single convention for coincident points, as the winding number for individual curves must be computed without any knowledge of the other components that make up the shape, which can be unintuitive if that shape is itself closed. Nevertheless, we can evaluate the winding number at a coincident point for a single curve just as before, by computing the half-integer winding number of the closed shape and subtracting the contribution of the closing line.

This convention for a coincident winding number is well-justified in a mathematical context as well. As explained by [79], the generalized winding number can be understood in the context of ray casting as the *average* number of intersections from rays cast in all directions from q. When q is on a straight line, only two of the uncountably infinite possible directions will intersect the line, resulting in an average of zero intersections. Furthermore, the discontinuity in Equation 5.5 is, in fact, removable, and evaluating the remaining integrand agrees with this interpretation of coincident winding numbers.

Despite this conceptual clarity, evaluating coincident winding numbers is complicated in practice, as it is computationally expensive to perform the projection necessary to identify when a point is located exactly on a Bézier curve [103]. The exception to this is the endpoints, which are interpolated exactly by the first and last control points of the curve. At such points, the coincident winding number is equal to the signed angle spanned by the non-coincident endpoint, and a *tangent* vector at the coincident endpoint (see Figure 5.10). Furthermore, computing such a tangent is trivial at the endpoint of a Bézier curve, as it is defined by the endpoint and the adjacent control node. Interior points become endpoints in linear time by the repeated bisections of our algorithm. As this spanned angle must account for full revolutions of this span around the tangent vector, our algorithm only applies this edge-case to curves with simple and convex control polygons (see Algorithm 7). This particular criterion is typically met after very few bisections, and

ensures that no full revolutions can occur on the local subcurve.

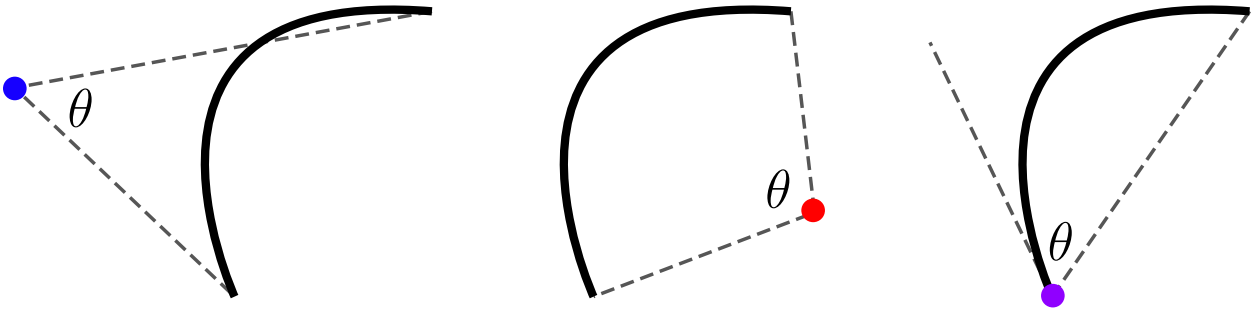


Figure 5.10: For points located outside the closed curve (left, center), the winding number is proportional to the angle subtended by the endpoints. For points located directly on an endpoint (right), we define the winding number as proportional to the angle subtended between the non-coincident endpoint and a tangent vector.

Algorithm 7: `ConvexEndpointWindingNumber`: Return the convention for "coincident winding numbers" as outlined in Section 5.7

Input: Γ: Rational parametric Bézier curve
q: Query point
Output: w_Γ: The winding number evaluated at the endpoint of the curve

if $q = \Gamma(0)$ **then**
  return $\frac{1}{2\pi} \times \left(\text{Signed angle subtended by } \overrightarrow{q\Gamma(1)} \text{ and } \overrightarrow{\Gamma'(0)} \right)$.
else
  return $\frac{1}{2\pi} \times \left(\text{Signed angle subtended by } \overrightarrow{q\Gamma(0)} \text{ and } -\overrightarrow{\Gamma'(1)} \right)$.

5.8 Numerical Experiments and Results

5.8.1 Robustness of Containment Queries on Curved Geometry

We first show the utility of a generalized winding number approach to containment queries in the context of messy CAD geometry. We compare our approach to conventional ray casting for containment, which remains the de-facto standard for exact containment in curved watertight geometry. Consider the geometry of Figure 5.11, which is composed of 87 linear segments and 477 cubic Bézier curves. We remove the marked curve in Figure 5.11 and determine containment at each pixel in the image using a simple ray casting algorithm that extends the ray to the right of the

pixel and counts the number of intersections with the shape. As expected, the use of a ray casting algorithm causes catastrophic issues, in the sense that the containment errors are located at a great distance from the actual geometric error. This makes the model unusable in the applications of interest without specific oversight and (often user-driven) correction of these errors.

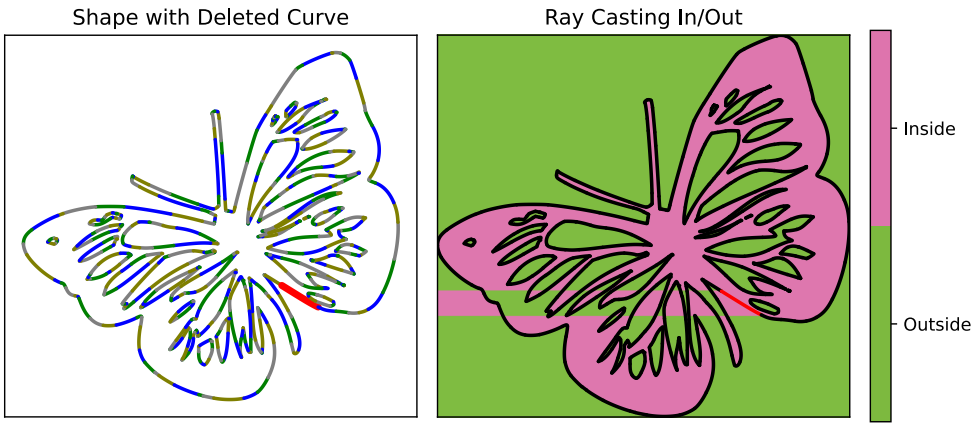


Figure 5.11: (left) A watertight geometric shape, with the exception of the curve in red that is removed during calculation. (right) Ray casting classifies points as interior (pink) and exterior (green). As expected, the geometric error leads to numerous errors in containment queries.

We then apply our generalized winding number algorithm to the same geometry in Figure 5.12, evaluating at each pixel the winding number field generated by the bounding geometry. In this example, points which we expect to be interior to this shape have winding numbers close to 1, while the value for points we expect to be exterior is close to 0. We see that deleting the curve does degrade the field, but that the most severe effects are *localized* to the site of the geometric error. In some sense, the locality of this degradation is unintuitive, as containment is necessarily a *global* property of points relative to bounding geometry, i.e., the winding number at every point is dependent on every curve. Importantly, however, the influence of distant curves becomes rapidly negligible. While the exact value of the winding number now a non-integer value (nearly) everywhere in the domain, away from the deleted curve the difference from the nearest integer is nowhere large enough to have an adverse effect on the actual determination of containment.

A natural mapping of winding numbers to containment classifications is to round each value

to the nearest integer and apply a non-zero rule. For example, in Figure 5.12 we see that the *rounded* winding number produces a clear boundary between regions along the 1/2 isocurve. While this implied boundary lacks the curvature of the deleted curve, it is a reasonable approximation and nevertheless allows for the surface to be used immediately in applications. This behavior is especially desirable when geometric errors arise from small gaps between components of the mesh, as the resulting containment query reflects the simplest closure of the shape permitted by the provided geometric data.

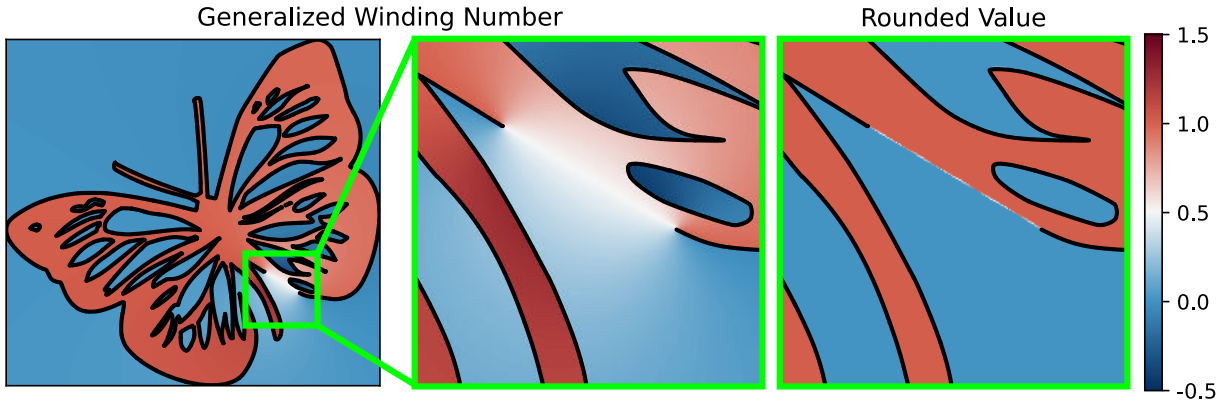


Figure 5.12: (left) The generalized winding number computed using Algorithm 3. (center) The winding number in the region around the deleted curve degrades smoothly. (right) Rounding the winding number produces an approximation that better conforms to the designer's intuition.

Through Figure 5.13, we reiterate the two important kinds of robustness in this work. The shape in this example is intentionally deformed such that no pairs of adjacent edges are connected. From a distance, i.e., the scale at which the model geometry is visually "good enough," there are functionally no irregularities in the shape's appearance. This mirrors the realistic setting, in which such imperceptible tolerances are unexpected and remain catastrophic for conventional containment algorithms. As before we apply our algorithm to each pixel in the bounding region, and plot the winding number field alongside its difference from the nearest integer.

The first type of robustness is derived from our specific computation of the generalized winding number through Algorithm 3, which is completely robust to non-watertight geometry. Because we only ever consider individual curves without any reference to their overall arrangement, both

the computational performance and accuracy of our calculation is completely unaffected by the non-watertightness of the shape. Furthermore, our adaptive algorithm ensures that the winding number is computed stably at arbitrary distances from individual curves. On the other hand, *any* containment query derived from generalized winding numbers is robust to geometric artifacts in the boundary. This is because although even small errors have far-reaching influence on the scalar field of winding numbers, this influence degrades rapidly and gracefully away from the sources of these errors. Even at this frequency of topological errors in this example, the resulting values of the field still very clearly partition the shape into the expected interior and exterior.

It is noted in [79] that a useful perspective on the fractional value of the winding number is as a measure of confidence for the derived containment query. For example, points for which the winding number is close to an integer can be considered "more likely" to be classified accurately according to the unknown, but presumably watertight shape that lacks any of the present boundary artifacts. On the other hand, points with winding numbers near half-integer values are often close to the boundaries that are only implied by the rounding heuristic, and correlate with containment classifications that are less informed by existing surface elements. In Figure 5.13, we specifically identify each point in the image with a winding number in the range [0.25, 0.75], which are the only points that we can therefore consider to have an "uncertain" classification according to our rounding heuristic. From this, we can see that the *vast* majority of points in this image are classified with high confidence.

Of course, the rounding heuristic is not the only way one can determine containment from a fractional winding number, as access to the full winding number field through our algorithm permits treatment of containment through standard segmentation strategies. For example, [79] meshes the interior of the shape, utilizing an energy minimization technique to enforce additional smoothness in the resulting segmentation. Additionally, the above usage of the winding numbers as a measure of confidence can be made more rigorous through further statistical analysis. In [151], the generalized winding number field is used as a prior for a number of statistical distributions, including likelihood of containment. This rigor is desirable for certain applications, as the fractional

value of the winding number alone cannot be interpreted as a probability for the accuracy of the derived containment query. Because we are ultimately concerned with efficient evaluation of the winding number in this work, we use only the rounding heuristic to determine containment in our examples.

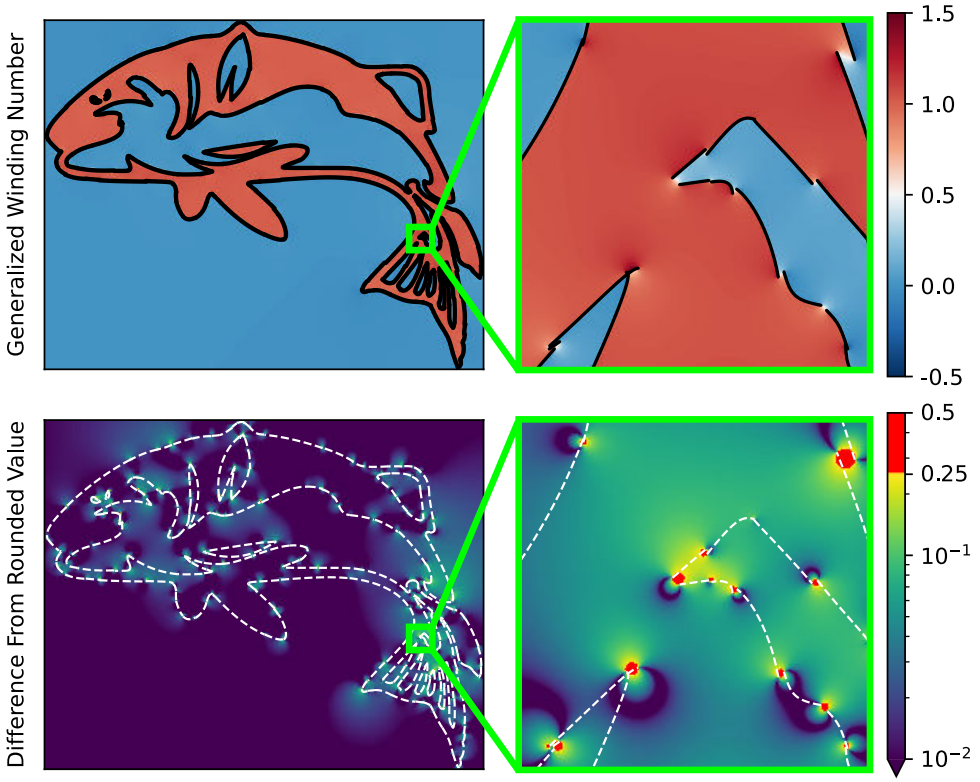


Figure 5.13: (top) Generalized winding number for a shape that is intentionally deformed to slightly separate adjacent curves. (bottom) Absolute difference (log scale) between computed winding number and rounded winding number. Points for which the difference is greater than 0.25 are highlighted, as this indicates some degree of uncertainty in the classification. As we see, such points are very sparsely distributed throughout the domain.

5.8.2 Algorithm Performance

We now consider the computational performance of Algorithms 3 and 4. In each of the following examples, we evaluate our algorithm in the context where exact geometric fidelity is prioritized, and set all numerical tolerances to zero. For comparison, we consider state-of-the-art techniques that have been adapted to collections of parametric curves. Perhaps the most common

winding number-based approach for watertight geometry would be to discretize the shape into approximating polygons with linear edges and to apply a standard point-in-polygon algorithm to the polygonal shapes to determine containment. If watertightness cannot be guaranteed, the generalized winding number approaches of [79, 175] admit a straightforward adaptation to a linearized shape.

In demonstration of this fact, consider Figure 5.14, where we compare the performance of our serial implementation of the proposed adaptive strategy to such a linearized method. Specifically, we consider refinements of each curves in the shape into a fixed number of linear segments. We note that the shape considered is non-watertight and non-manifold, making it a particularly challenging case for conventional containment queries. Nevertheless, the fractional winding number field clearly partitions the space according to an intuitive understanding of its interior.

By comparing wall-clock runtime, we see that the use of a fixed level of refinement of the curve leads to a more efficient calculation of the generalized winding number as expected, as curve subdivisions can be computed during a preprocessing step and reused. While the analogous overhead in the adaptive algorithm can be similarly mitigated through hash maps that cache the results of curve subdivisions, we consider further implementation optimizations for collections of curves to be future work, and instead focus here on efficient and accurate calculation for individual curves. Indeed, when geometric fidelity is not a concern, using a fixed linearization to compute generalized winding numbers with Algorithm 3 is a reasonable approach, although it may involve a considerable memory footprint for larger models.

Importantly, such a linearized method offers no guarantees for geometric fidelity. By comparing the level of refinement against an approximate likelihood that points are ultimately misclassified, we see that although the rate of misclassification decreases with increased linear refinement, it does so at a rate that we consider to be far too slow to be practical for many applications of interest, especially for downstream applications that are sensitive to *any* misclassifications. This is also an issue for methods that sample the bounding geometry non-deterministically, such as those of [123, 10, 144]. In contrast, the proposed method is able to achieve perfect geometric fidelity for points that are arbitrarily close to the boundary. In place of these fixed linearization methods, we

turn our attention to those that operate directly on curved geometry without approximation.

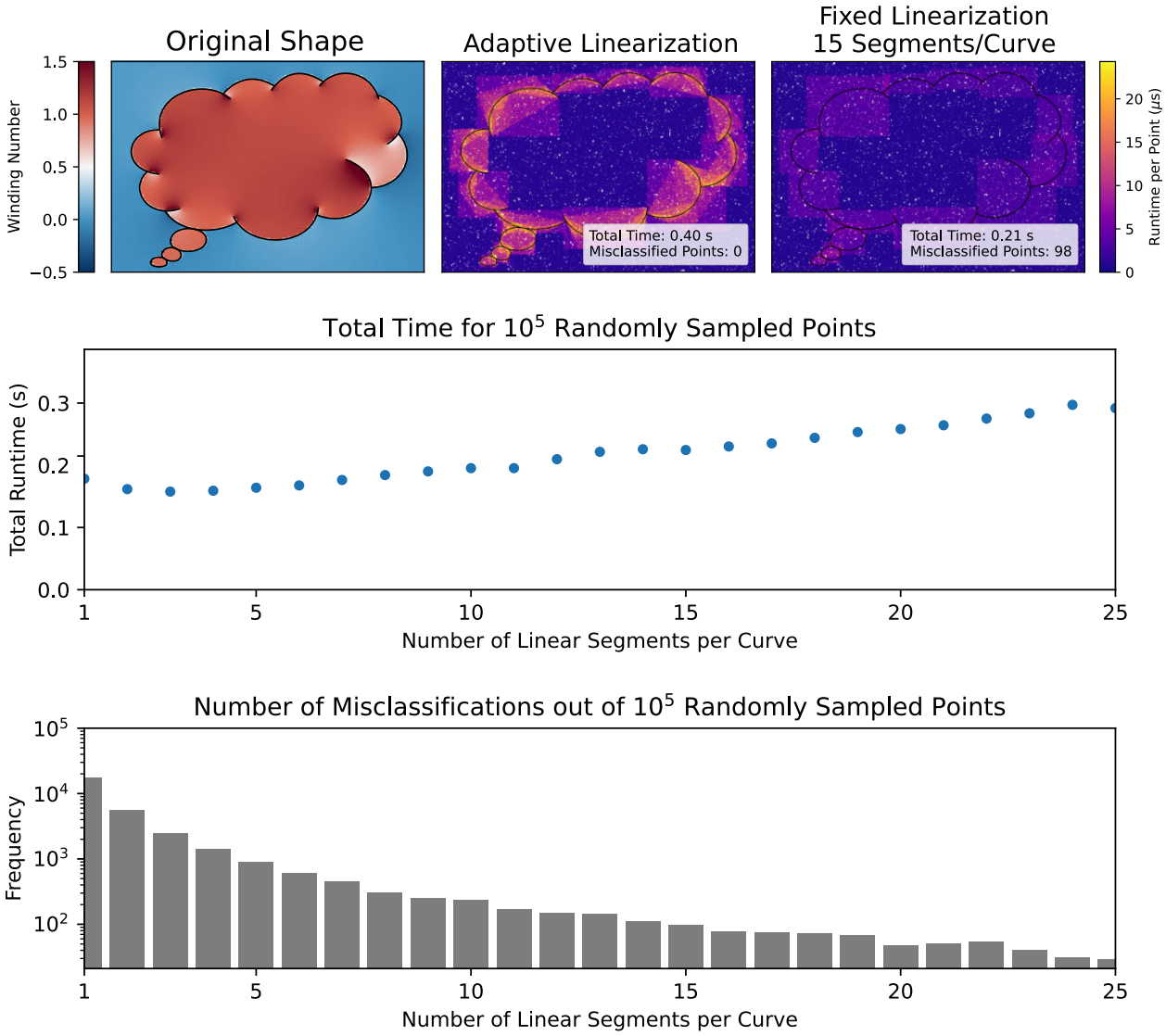


Figure 5.14: Given a non-watertight, non-manifold shape, we observe the practical effect of curve linearization on geometric accuracy and runtime. We compare our adaptive strategy to those that uses a uniform, piecewise linearization of the curve at fixed levels of refinement. We record the total time spent to evaluate the generalized winding number for 10^5 points randomly sampled from a uniform distribution, and the number of these points which are ultimately misclassified. We see that a fixed linearization can offer better computational performance, as subdivisions of the curve can be efficiently precomputed. However, we note that even at high levels of refinement there are still a considerable number of misclassifications, which is unacceptable in many downstream applications of interest.

There remains the issue that the methods known to the authors that operate directly on curved geometry à la Bézier clipping [120] can do so only in the watertight case. This regrettably

means that no direct comparison for the performance Algorithm 3 can be made at this time. In its place, we note that our procedure for computing the exact generalized winding number for any open, curved object requires applying an *arbitrary* integer winding number algorithm on the closed shape and subsequently subtracting out the contribution of the closure of the curve. It is in this context that we demonstrate the performance of Algorithm 4, which is one such algorithm that computes the integer winding number of a closed, curved object. We show that Algorithm 4 is considerably more performant than alternatives utilizing crossing numbers for containment queries on closed, curved geometry, as such methods necessarily introduce inefficiencies through a reliance on ray casting.

We first consider a common procedure for ray casting that applies a geometric *binary search* to compute ray-Bézier curve intersections [46]. In this algorithm, we extend from the query a ray in the direction of the closest edge of a bounding box, and recursively bisect the curve until guarantees for potential intersections can be made. If a subcurve is sufficiently linear (see Algorithm 5) such that an intersection is guaranteed, then the signed intersection can be recorded. If the ray does not intersect the subcurve's bounding box, then it can be discarded. This procedure converges to points of intersection linearly, and is capable of identifying multiple intersections with the same curve.

Bézier clipping [120, 150] is a similar procedure, discarding sections of the curve that are guaranteed to have no intersections with the ray. However, the convergence of this approach to points of intersection is quadratic, as the curve is instead split into three subcurves at each iteration.

To make the appropriate comparison to Algorithm 4 in the context of exact winding numbers for non-watertight geometry, we utilize each *within* the framework described in Algorithm 3. That is to say, we iterate over each open curve in the model and close it, compute the integer winding number with Algorithm 4 or one of the above alternate techniques, and subtract out the contribution of the closure. We note that, while these alternate techniques themselves are well-known in the context of ray-tracing and containment of closed shapes, computing generalized winding numbers in this way is, to our knowledge, itself a novel application of their use.

In comparing these three algorithms, there are a number of specific implementation details that govern the wall-time of their evaluation, in particular the use of a spatial index to efficiently handle far away points. However, we are concerned with their efficiency in the near-curve regime, where the number of *curve evaluations* (most commonly occurring as a result of a curve bisection) is an effective proxy for computational efficiency between the methods. Furthermore, the computation of the exact fractional value for the generalized winding number only involves the evaluation of a single arccosine for each curve in the model, and this cost is equivalent across all three methods.

Our first example in Figure 5.15 uses a relatively simple shape featuring at most polynomial, cubic Bézier curves. To compare the three algorithms (binary search, Bézier clipping, and the proposed approach presented in Algorithm 4), we randomly sample 250,000 points from a bounding box around the shape using a uniform distribution, and count the number of curve evaluations on each that needs to be made to determine containment with perfect geometric fidelity. Again, we emphasize that all three methods are being used here to compute the generalized winding number, which they do by computing the integer winding number of the closed shape for each curve in the model.

As expected, the binary search ray casting algorithm performs the poorest, with a considerable number of points requiring more than 15 curve subdivisions to make an evaluation. This is because the terminating condition of the naive algorithm is linearity of the curve component near the intersection, for which the number of bisections increases with curvature. This behavior is improved considerably with Bézier clipping, which performs more sophisticated subdivision (at greater computational expense) by considering the convex hull of the curve. However, the proposed algorithm improves on this further, with no points requiring more than 8 function evaluations, and most requiring only one or two.

Furthermore, we recognize that for the vast majority of points, *zero* function evaluations are required for each algorithm. This is because the first containment check performed in each is one to an axis-aligned bounding box for the entire curve, and most points lie outside the bounding box for most curves. For this reason, integer winding numbers for most combinations of points and

curves are known to be zero without any curve evaluations.

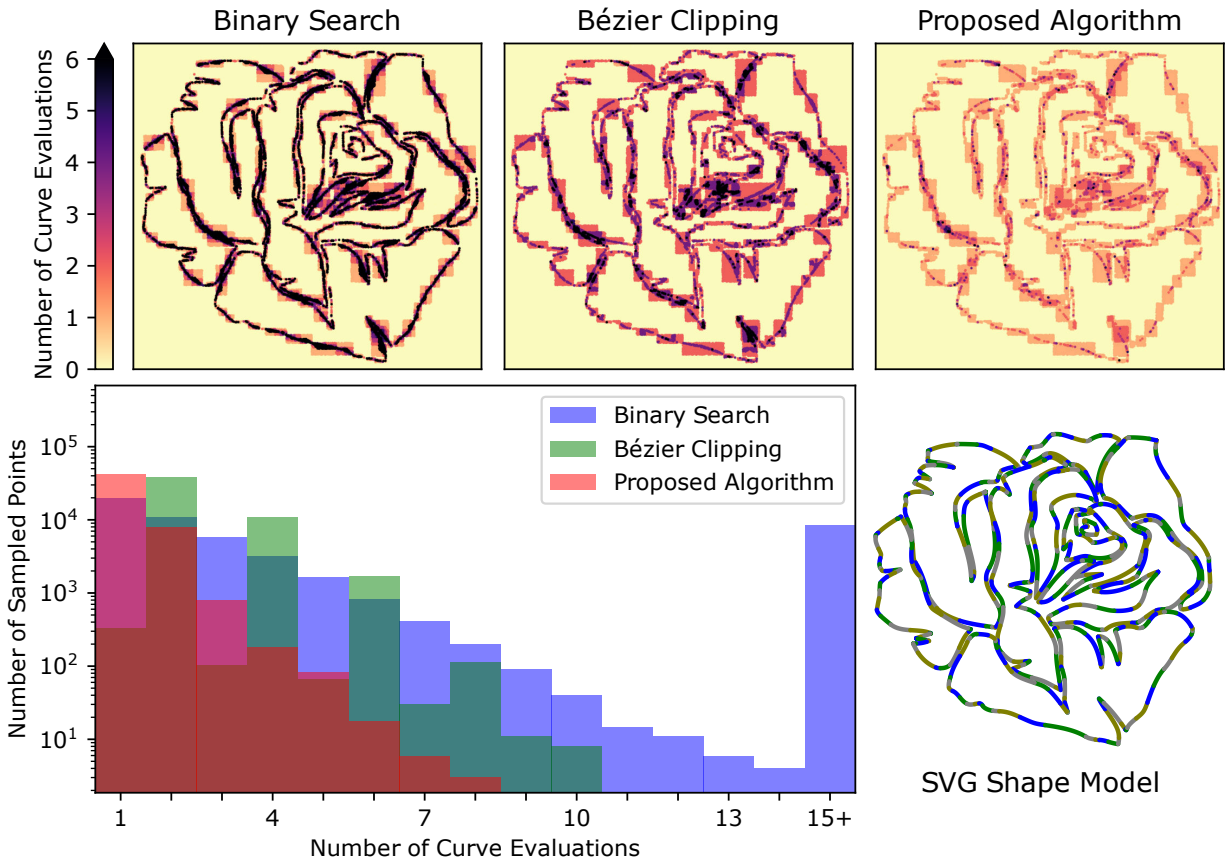


Figure 5.15: We compare the cost of a conventional geometric bisection method, Bézier clipping, and our proposed method on a watertight model. We uniformly sample 250,000 points from a bounding box and measure the number CAD model evaluations, i.e., curve subdivisions, that are necessary to evaluate the winding number at each point to full geometric precision. In the histogram, we omit the number of sampled points that require zero function evaluations, as this occurs whenever the point is outside each curve's bounding box, occurring at the same frequency for each method.

One explanation for the relatively poor performance of the two ray casting algorithms is that such procedures are unnecessarily informative. Not only do ray casting algorithms *identify* potential intersections, but they also inherently *locate* the intersections, information which is rarely needed when the ray itself is chosen arbitrarily. In contrast, by not needing to compute the location of these intersections, the proposed algorithm is able to converge with considerably fewer curve evaluations, despite only doing so linearly. This is particularly relevant in the case where the curves of interest contain multiple overlapping components, self-intersections, or regions of significant curvature, the

latter of these cases being tested directly on a rational curve in Figure 5.16. In these examples, the ray casting algorithms perform particularly poorly, as any ray extended will intersect the curve multiple times, incurring additional cost with each.

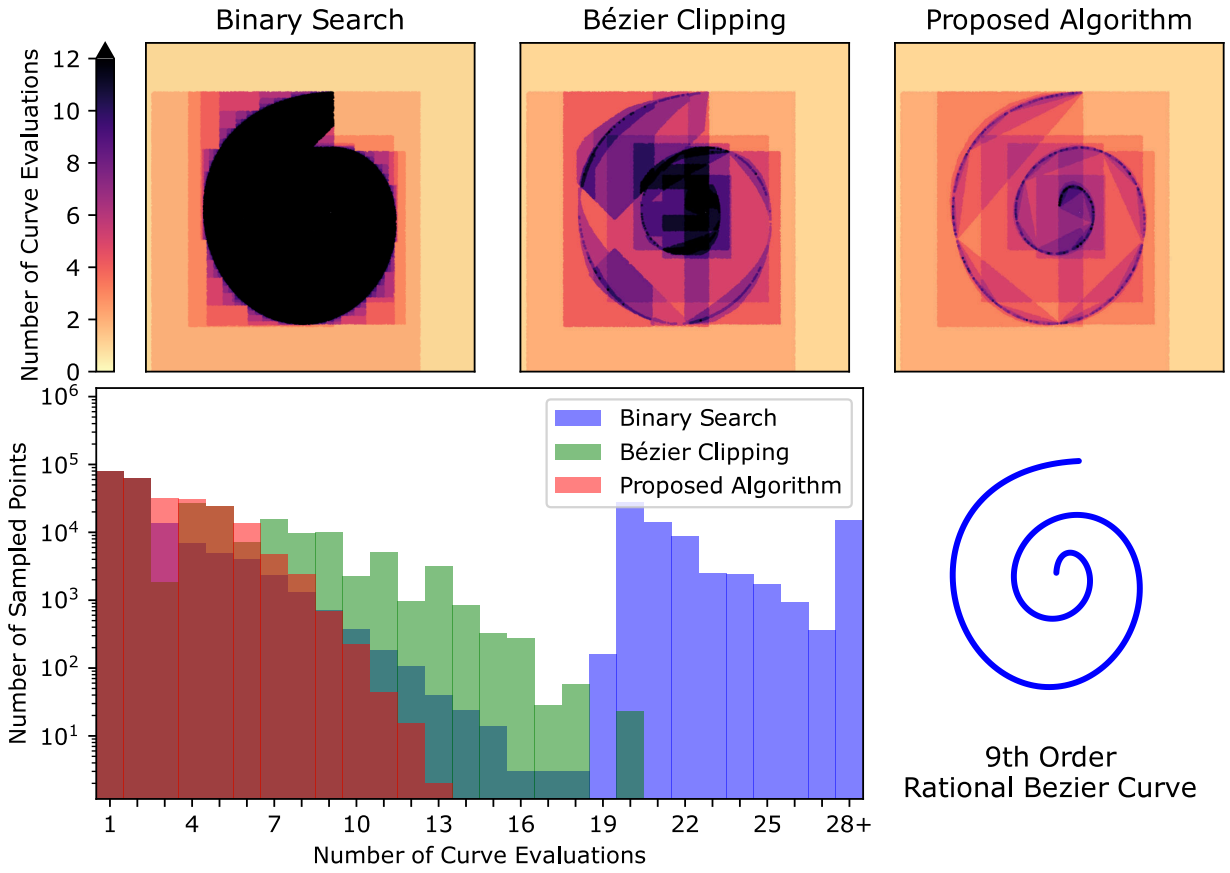


Figure 5.16: For higher-order curves, additional curve subdivisions are necessary to reach perfect geometric fidelity. The proposed algorithm still has superior performance in such cases, as the location of the various intersections need not be computed directly.

Finally, we consider the sensitivity of our procedure for computing generalized winding numbers to the numerical tolerance introduced in Algorithm 5. We do this by considering performance and accuracy as a function of this tolerance, respectively measured in terms of the number of curve evaluations required to determine containment and the number of misclassifications caused by the final layer of implied linearization. These metrics are evaluated over 10^5 points sampled from a bounding box for the closed shape in Figure 5.17, which for demonstration purposes depicts a 9^{th}

order, rational curve that far exceeds the complexity of those typically used in practice. We see that invoking Algorithm 5 to prematurely terminate the procedure does marginally reduce the overall computational cost, but that this performance improvement is closely correlated with losses in classification accuracy. This effect is amplified on shapes with such complex curvature, as it becomes more likely that the implied linearization imposed by the tolerance will produce misclassifications. At the same time, we observe that even for very intricate shapes, the performance of Algorithm 4 rapidly stabilizes along both metrics. As a result, our practical preference is to set this tolerance to zero even in cases where geometric accuracy is not a priority.

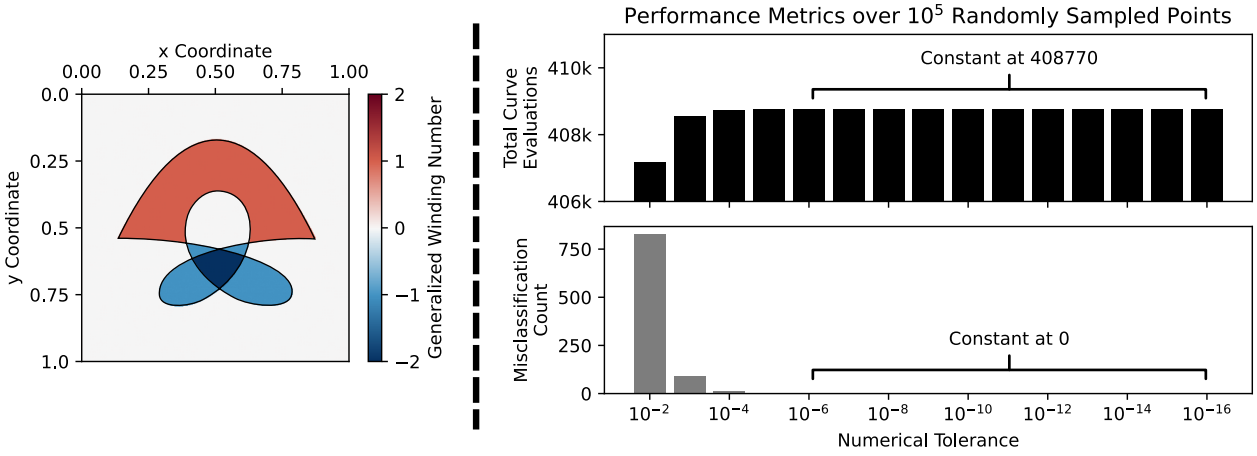


Figure 5.17: Beyond a certain threshold, the adaptive nature of the proposed algorithm causes it to be insensitive to the numerical tolerance in Algorithm 5. In practice, we set this tolerance to zero, as this ensures exact geometric fidelity while not incurring a large additional computational burden.

As is always the case when generalized winding numbers are used to determine containment, there is still an assumption that the collection of curves are reasonably *oriented*. Because the scalar field generated by a single curve is computed completely independently of every other curve by design, a single reversed curve interferes with the contribution of surrounding curves, functionally reversing nearby containment queries, as can be observed in Figure 5.18. While the induced classification errors are still local to the geometric error, and would no doubt cause catastrophic errors *without* the use of generalized winding numbers, they nevertheless represent a type of geometric error unaccounted for by the current method.

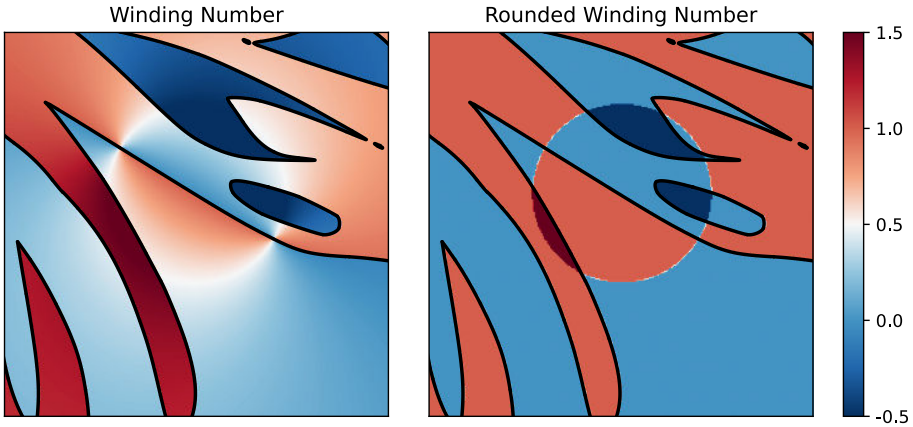


Figure 5.18: GWN field generated by the shape in Figure 5.12, but with the orientation of the highlighted curve reversed rather than removed entirely. The reversal of this curve causes surrounding containment classifications to be swapped within a local region.

5.9 Concluding remarks

In this paper, we proposed a generalized winding number algorithm for collections of parametric curves, addressing the inherent challenges for curved geometry that are not shared by their linear counterparts. For the vast majority of points that are considered "sufficiently far" from a given curve, we circumvent the need for expensive linearization of the shape by treating the curve directly as a single segment connecting its endpoints. For all other points, we demonstrated that our recursive algorithm outperforms existing ray casting algorithms adapted for this context. Finally, we formalize a procedure for handling winding numbers of points that are coincident with the curve, further increasing the practicality of our algorithm. In each case, this algorithm makes minimal reference to the often near-singular integrals from which the winding number is theoretically derived, instead operating strictly according to geometric and trigonometric principles. Nevertheless, we are able to *exactly* compute generalized numbers for curved shapes in two dimensions. Whereas the focus of this work has been on efficiently and robustly computing winding numbers for individual curves within a collection, follow-up work will consider accelerating the overall query workflow using a spatial index, as in [79, 10, 180] and exploiting the inherent parallelism within Algorithm 3 via threaded and/or GPU implementation.

Chapter 6

Generalized Winding Numbers for Trimmed NURBS Surfaces

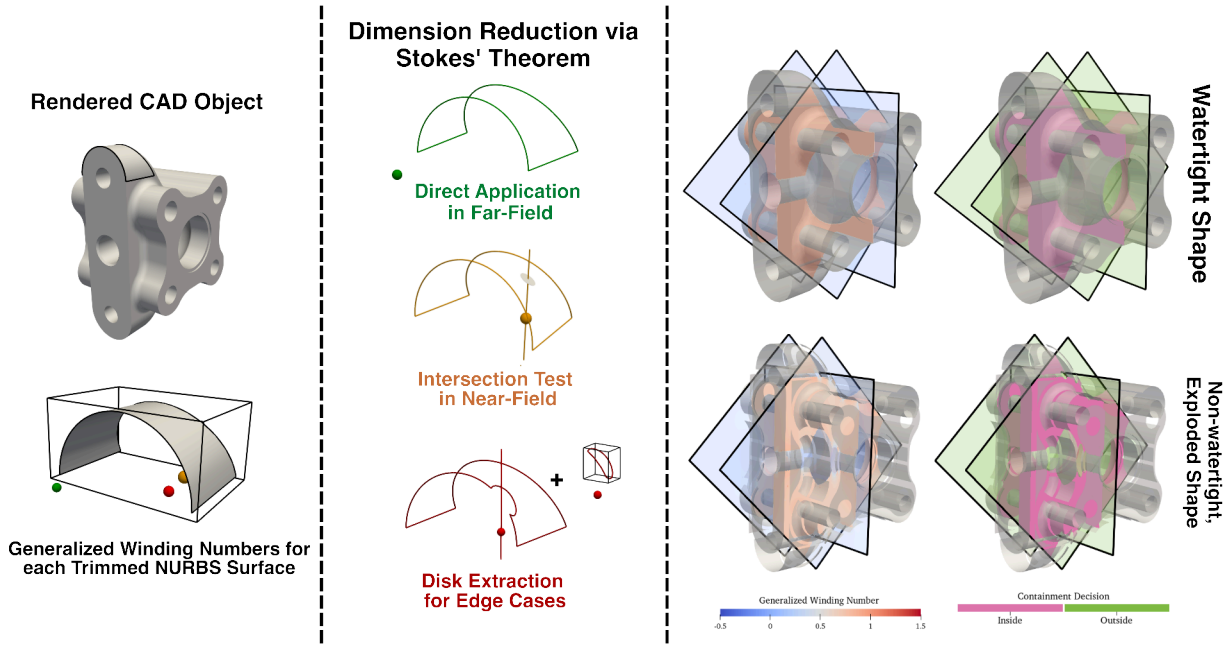


Figure 6.1: CAD models composed of trimmed NURBS surfaces are ubiquitous in both engineering and design, but robustly defining an interior for such objects can be challenging in the presence of imperceptible gaps or overlaps between adjacent patches in a model. We present a method of evaluating the field of *generalized winding numbers*, which can be used to define a containment query that is indifferent to the watertightness of the model. Our algorithm uses Stokes' theorem to reformulate the problem as a 1D line integral over the boundary of each patch, which we solve using an adaptive quadrature technique that is accurate to user tolerance. In circumstances where Stokes' theorem is otherwise theoretically unjustified (i.e., near to the surface), we perform a line-surface intersection test to recover the computational advantages of our boundary formulation. Edge cases for query points nearer to surface boundaries are handled accurately and robustly by extracting a parametric disk from the surface and are processed separately.

In collaboration with Kenneth Weiss

6.1 Abstract

Efficient and accurate evaluation of containment queries for regions bound by trimmed NURBS surfaces is important in many graphics and engineering applications. However, the algebraic complexity of surface-surface intersections makes gaps and overlaps between surfaces difficult to avoid for in-the-wild surface models. By considering this problem through the lens of the generalized winding number (GWN), a mathematical construction that is indifferent to the arrangement of surfaces in the shape, we can define a containment query that is robust to model watertightness. Applying contemporary techniques for the 3D GWN on arbitrary curved surfaces would require some form of geometric discretization, potentially inducing containment misclassifications near boundary components. In contrast, our proposed method computes an accurate GWN directly on the curved geometry of the input model. We accomplish this using a novel reformulation of the relevant surface integral using Stokes' theorem, which in turn permits an efficient adaptive quadrature calculation on the boundary and trimming curves of the model. While this is sufficient for "far-field" query points that are distant from the surface, we augment this approach for "near-field" query points (i.e., within a bounding box) and even those coincident to the surface patches via a strategy that directly identifies and accounts for the jump discontinuity in the scalar field. We demonstrate that our method of evaluating the GWN field is robust to complex trimming geometry in a CAD model, and is accurate up to arbitrary precision at arbitrary distances from the surface. Furthermore, the derived containment query is robust to non-watertightness while respecting all curved features of the input shape.

6.2 Introduction

In computer graphics and adjacent fields, 3D objects are often defined exclusively in terms of a boundary representation (B-Rep). While these B-reps are very useful for flexibly designing 3D objects within a CAD system, in some sense they only tenuously define a legitimate interior volume. For example, accessing the interior of such an object requires explicitly defining from the B-rep

a *containment query* that can determine whether a given point lies inside or outside the object. Particularly in the context of general curved surfaces, this containment query is often iterative and non-trivial to implement robustly and efficiently.

In this work, we consider B-Reps composed of a (possibly unstructured) collection of 3D non-uniform rational B-spline (NURBS) tensor product surfaces. Many design systems allow these rectangular patches to be trimmed so that they describe a wider variety of shapes. These trimming curves are themselves typically defined by a (again possibly unstructured) collection of 2D NURBS curves in the parameter space of the surface.

While evaluating containment queries on trimmed NURBS surfaces is a generally difficult task, in this work we focus on the compounding difficulty of performing containment queries on B-Reps which are not necessarily *watertight*, in the sense that collection of surface patches may not form a mathematically closed volume. In graphics-oriented tasks where visibility is the primary concern, non-watertightness is a natural consequence of holes within a patch, gaps between patches, or even more general non-manifold edges.

There are many applications in which a well-defined volumetric interior is a pre-requisite, particularly those in animation or engineering contexts [112, 189]. Even so, B-Rep models designed specifically for these purposes can be unexpectedly non-watertight, sometimes due to simple human error in the design process, or more often due to differing tolerances between applications. In the context of trimmed NURBS surfaces, however, such gaps are inevitable even in well-designed models due to the complex nature of surface-surface intersections [149]. In other examples, the use of Boolean operations during the construction of a model can introduce problematic non-manifold edges. We show examples of each kind in Figure 6.2. In this and other examples throughout this work, we use CAD models from the **ABC Dataset**, a collection of CAD models of varying quality used to benchmark various algorithms in computer graphics and machine learning [91].

A potential solution is to restore model watertightness directly via surface repair. While direct repair of the model within a CAD engine to close gaps and/or clip overlaps is possible, such manual correction is often time-consuming and error-prone. Automated systems for surface

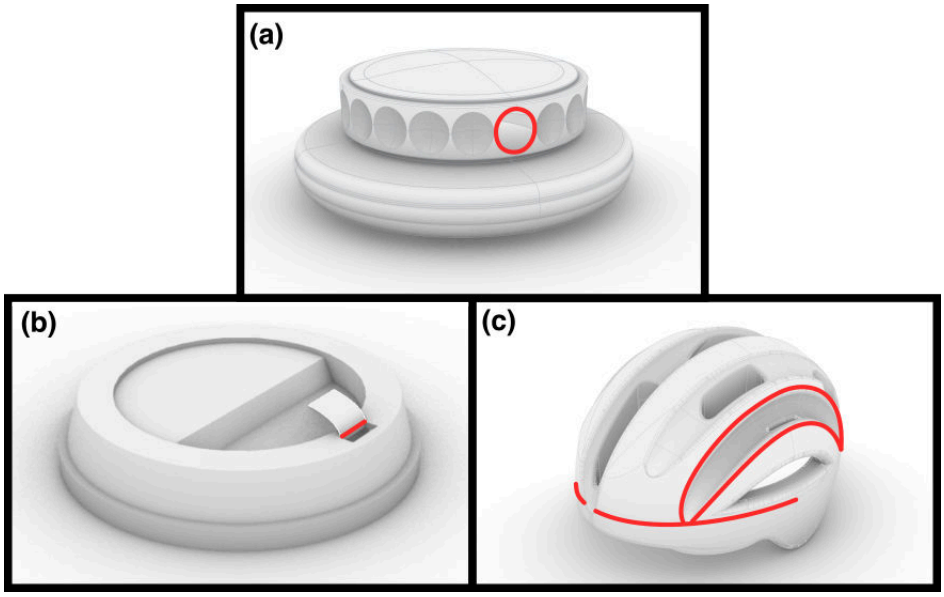


Figure 6.2: We demonstrate common ways in which a B-Rep model defined by NURBS surfaces can be non-watertight, such as through the presence of (a) holes, (b) non-manifold edges, (c) edges that visually appear to be matched to other patches, but could become non-watertight when ported between systems with differing tolerances.

repair are typically applied globally during the export of CAD models as a part of a larger mesh-generation pipeline [161, 105]. However, the focus of such work is often the creation of watertight models composed of *linear* elements (e.g., triangles), and is still an area of active research for arbitrary curved and trimmed surfaces.

Indeed, discretization of individual trimmed surfaces *into* a linear mesh permits immediate application of not only more traditional surface repair techniques, but also the growing body of work that defines containment queries for non-watertight shapes [79, 10, 48]. However, there are many contexts in which this type of approximation through discretization would either be too costly to capture complex geometry, or has the potential to introduce unforseen and otherwise unacceptable errors in downstream applications. In this case we take multiphysics simulation as an important class of examples, and point to the work of Sevilla et al. for a comprehensive review of the complications that arise when a piecewise linear boundary is used to approximate a curve [154, 155, 156].

In the context of finite element analysis specifically, this desire for exact geometric fidelity has

led to the development of techniques that work directly on boundaries defined by CAD geometry such as in isogeometric analysis [73, 111]. Particularly relevant to the proposed computational techniques are methods of immersogeometric analysis, in which fluid-structure interactions are modeled by placing a complicated CAD model on a simpler computational grid on which the numerical solution is computed [87, 52]. In such methods, it is necessary to know whether points that are structured with respect to this background grid are inside or outside the CAD model, which can be particularly challenging for points located arbitrarily close to the B-Rep.

In total, we seek to extend the capabilities of these and other techniques by introducing an evaluation method for containment queries with respect to a non-watertight B-Rep of trimmed NURBS surfaces. We accomplish this through the *generalized winding number* (GWN) [79], a mathematical construction that is useful not just in defining such a query, but in a variety of other geometry processing routines as well, such as Boolean operations [175], surface reconstruction [79], and others. Specifically, we present in this work the following principal contributions:

- We extend the theory of generalized winding numbers to the context of trimmed NURBS patches by defining and evaluating the 3D GWN field with respect to such objects. As a result, we achieve a containment query for B-Reps composed of such shapes that is robust, and indeed indifferent, to watertightness.

- We describe a novel algorithmic framework for evaluating the GWN with respect to a trimmed NURBS surface by reformulating the relevant surface integral as a line integral along the surface boundary via Stokes' theorem. Doing so greatly improves numerical stability for the integrals, which we pair with an adaptive quadrature scheme to achieve results up to arbitrary accuracy.

- We accurately evaluate the GWN for points which are arbitrarily close and even coincident with the surface by directly identifying and compensating for the jump discontinuity in the GWN field over surfaces. As part of this process, we robustly handle visibility queries in the 2D parameter space of each surface with the 2D GWN. This permits evaluation of

the GWN field both in the "far-field" and "near-field" of the NURBS surface without any discretization.

Practical applications of the GWN to large-scale B-Reps typically use a spatial index to accelerate queries (e.g., [79, 10]), which we intend to pursue in follow-up work. Instead, our focus in this work is on exact and efficient evaluation of the GWN with respect to individual trimmed NURBS patch, the cost of which scales nearly linearly with the number of trimming curves. Our algorithm has been implemented in Axom, a BSD-licensed open-source library of Computer Science infrastructure for HPC applications [23].

6.3 Background and Related Work

6.3.1 Containment Queries in 3D CAD Applications

While containment queries are a fundamental operation for many CAD applications, methods of their implementation can vary considerably, especially when dealing with curved geometry.

As in 2D, the typical containment query is built upon the ray-casting method. In the 3D problem context, there is considerable overlap between this process and the adjacent task of ray-tracing for the purposes of rendering, which typically require determining the first intersection of a ray with a surface, but an important distinction is that ray-casting for containment queries permits an arbitrary direction of the cast ray.

Although much research focus in ray-tracing relates to triangulated surfaces, as many hardware platforms are capable of calculating ray-triangle intersections massively in parallel [65, Chapter 3], there is also interest in applying such techniques directly to trimmed NURBS surfaces [183, 147, 146]. Of particular note are recent methods for ray-tracing bilinear patches, as, similar to triangles, they admit direct geometric formulae for computing their intersections with rays [140].

On more general curved surfaces, the surface-line intersection routines used to determine containment are typically more complex and iterative. For example, algebraic methods find solutions in terms of the parametrization of the ray and/or surface, either through numerical root-finding [110],

convex optimization [108], or even via singular value decompositions of matrix embedding problems [159]. These algebraic methods can often find intersection points to a high degree of numerical precision, but at the cost of additional sensitivity to the initial configuration of the relevant solver. On the other hand, geometric subdivision methods successively trim the surface at points where intersections are not possible, resulting in useful guarantees for their accuracy [120, 43].

However, the shortcomings of ray-casting in 2D persist in 3D as well, as the result remains highly sensitive to the specific configuration of surface and cast ray, which can lead to inadvertant misclassifications in watertight cases, and complete inapplicability for non-watertight shapes.

6.3.2 Generalized Winding Numbers in 3D

As described in the 2D problem context in the previous chapter, winding number methods are far more robust to common issues when determining containment in 3D shapes, with their non-generalized form being long understood as an alternative to ray-casting polyhedral shapes that is less sensitive to the choice of ray [24]. Indeed, the generalized winding number itself was first conceptualized specifically for the purposes of efficient evaluation of 3D point containment queries on messy STL geometry via a hierarchical spatial index and divide-and-conquer algorithm [79]. This work was later extended to entirely disconnected triangle soups in [10], improving performance with an error-controlled approximation of the GWN with respect to groups of mesh elements that are far from the query point.

Evaluation methods for the GWN field defined by triangle meshes and soups are particularly computationally efficient, as the solid angle which defines the 3D GWN can be directly and exactly computed for polygonal shapes. However, if the input surface geometry is not a triangle mesh, a naive application of these methods necessarily uses some geometric approximation of the surface to obtain the necessary input type, which in turn causes the GWN field of the approximation to diverge from the ground truth. At the same time, generating a tesselation of a NURBS surface is a non-trivial task, and high-resolution meshes can be prohibitively expensive to generate and store for complex models [183].

At the time of writing, there are no methods in the literature for evaluating the GWN field at arbitrary points relative to curved input geometry, particularly among methods which operate directly on the surface without any intermediate discretization.

For example, the work of [10] also describes a method of defining the field of GWN for oriented point cloud data, which could be sampled directly from a CAD-derived surface. Indeed, the work of [8] describes a method of doing so for the purpose of multiphysics simulation, sampling points randomly from the CAD surface and using them to robustly define an interior volume for an otherwise messy shape. However, each point in the derived cloud is weighted by a surface area from a Voronoi tessellation of the ground-truth surface, where the computed weight may itself be an approximation depending on implementation. Nevertheless, even if the weights can be accurately pre-computed directly from the CAD model, they still represent a quantization of the original surface into discrete, approximated components.

In comparing our objectives to these existing methods, it is useful to distinguish between two types of inexactness in the computed GWN field. The first is *inaccuracy* in the GWN calculation, where the error in the GWN field causes the derived containment query to result in misclassifications. This is almost exclusively a concern for points near to the surface, particularly when the GWN is computed using a low-quality intermediate discretization of the surface. In such cases, the query point can be identified as on one side of the discretized surface while being on the opposite side of the true surface, resulting in an off-by-one error in the GWN field.

The second is *imprecision*, which in some sense is a concern unique to the 3D case. In 2D, it is necessarily true that for a fixed set of input queries, there is some discretization of the input geometry that will produce the correct GWN field at each query point. However, this is not the case in 3D — the increased geometric complexity of the surface boundary means that the GWN field of a discretized, open surface will always diverge from the GWN field of the original surface at some points, even near to the surface. This is a largely separate concern from the accuracy of the GWN field, as the typical rounding strategy used to map the GWN field to a containment decision means that even imprecise GWN fields are (for reasonable error levels) still mapped to the same

containment decision.

As another point of comparison, the method of [144] is capable of rapidly evaluating the GWN field while maintaining geometric fidelity, at the cost of some approximation and consistency in the evaluation. Specifically, this method solves certain classes of PDEs by means of random walks and projections, including those which implicitly define the generalized winding number. This method is *accurate* (in expectation) by considering surface geometry exactly as provided through these projections, but loses some degree of *precision* for individual queries.

More generally, methods which discretize the surface geometry for the purposes of evaluating the GWN field precisely for far-away query points often do so at the cost of accuracy nearer to the surface. In this work, we develop a method for evaluating the GWN field that is accurate at arbitrary points relative to the input surface, and maintains precision through error-controlled numerical methods, all in service of avoiding misclassifications in the derived containment query.

6.3.3 Evaluating the GWN for Curved 3D Geometry

As described in the previous chapter, winding numbers in 2D measure the number of revolutions that bounding curves make around a point, which, in the watertight case, is equivalent to the crossing number derived from ray-casting. While conceptualizing the 3D winding number as revolutions of the boundary around a point is somewhat unclear, one can also consider the 2D winding number to be a measure of the total, signed angle (a solid angle in 3D) subtended by the boundary at the query point (see Figure 6.3).

Recall the definition of the 2D GWN $w_C(q)$ at a query point q with respect to a curve C as the total angle subtended by the curve, normalized by a factor of 2π. This results in a simple integral of the (signed) differential angle $d\theta$,

$$w_C(q) := \frac{1}{2\pi} \int_{C-q} d\theta. \tag{6.1}$$

Written in Cartesian coordinates, this is equivalent to the line integral,

$$w_C = \frac{1}{2\pi} \int_C \frac{x \cdot \hat{n}}{||x||^2} \, dx. \tag{6.2}$$

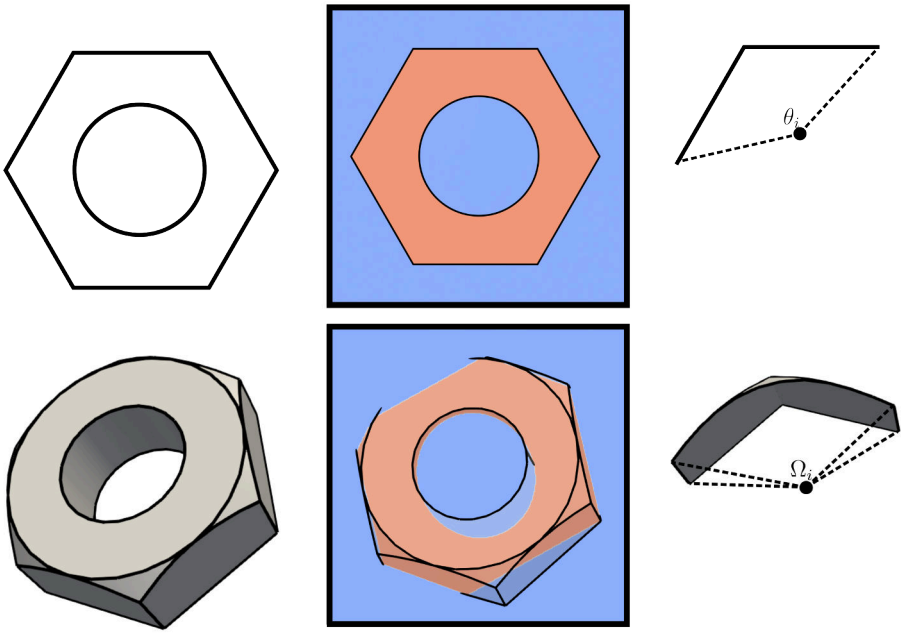


Figure 6.3: We show the integer (middle) and fractional (right) GWN for a simple 2D (top) and 3D (bottom) shape. In both 2D and 3D, the GWN of watertight shapes is integer valued, and the fractional GWN of non-watertight shapes is equal to the signed angle subtended by the shape boundary. To more easily visualize the 3D GWN as a scalar field, we consider a 2D slice of the field as it intersects with the surface.

To evaluate this integral directly via numerical quadrature is difficult, as the near-singular behavior of the integrand causes conventional quadrature methods to exhibit severe instability for points close to the curve. For linear elements with endpoints a and b, however, this subtended angle has a simple and direct formula

$$w_L := \frac{1}{2\pi}\arctan\left(\frac{||a \times b||}{a \cdot b}\right), \tag{6.3}$$

which can be leveraged in a scheme to compute the GWN of an *arbtirary* curve in 2D without the use of any quadrature scheme (See Chapter 5.

For the most part, this framework is unchanged for the 3D problem context. The 3D GWN $w_S(q)$ at query point q with respect to a surface S is defined as the total solid angle subtended by the surface, normalized by a factor of 4π:

$$w_S(q) := \frac{1}{4\pi}\int_{S-q} d\Omega, \tag{6.4}$$

which can be written in 3D as

$$w_S(q) = \frac{1}{4\pi} \iint_{S-q} \frac{x \cdot \hat{n}}{||x||^3} \, dS. \tag{6.5}$$

The same difficulties of numerical quadrature are also present, in some sense exacerbated by the necessity of comparably expensive tensor-product quadrature rules in evaluating the surface integral. The hope, then, is that this too can be avoided through an exact formula for the analogous linear element. Indeed, the GWN for a 3D triangle defined by vertices a, b, and c is given by

$$w_S := \frac{1}{2\pi} \arctan\left(\frac{|a \cdot (b \times c)|}{||a||\,||b||\,||c|| + (a \cdot b)||c|| + (b \cdot c)||a|| + (c \cdot a)||b||} \right), \tag{6.6}$$

But while any 2D curve can be closed with a straight line, only a trivial subset of 3D surfaces can be exactly closed by a collection of triangles. For more general 3D surfaces, the closing surface must itself be a high-order surface [149], and so computing the GWN of the closing surface (for the purposes of subtracting the value from an integer winding number of the closed surface) is typically just as difficult as computing the GWN of the original.

It is a known characteristic of the 3D solid angle from which the GWN field is derived that it can, in certain contexts, be defined entirely from the boundary of the surface. For example, perhaps the closest adaptation of the method of Jacobson et al. [79] to a 3D curved surface S would be to discretize *only* the boundary of the surface by $\partial S'$ (with $\partial S' \approx \partial S$), and construct a simple triangulated closure $\overline{S'}$ whose GWN field can be computed exactly. For query points which are far from the original surface, this approach is somewhat reasonable: although some precision is lost in the discretization, the GWN field of the approximated closure is related to the GWN field of the original surface for far-away points through the simple formula $w_S = w_{\overline{S}} \approx -w_{\overline{S'}}$. However, ensuring the accuracy of such an approach becomes problematic for query points near the surface. As in the 2D case, the equivalent relationship between w_S and $w_{\overline{S}}$ depends on the accurate calculation of the integer $w_{S \cup \overline{S}}$ at the query point, which still requires consideration of the geometry of the original surface. As we will see, our proposed method requires no such discretization of the boundary.

Altogether, this means that computing an accurate GWN for an arbitrary 3D surface by means of Equation 6.5 fundamentally requires the use of numerical quadrature to compensate for the unavoidable presence of curved geometry.

There are a number of existing quadrature techniques in the literature that can be immediately applied to Equation 6.5. For example, surface integrals over untrimmed patches can be evaluated using simple tensor-product quadrature rules. These tensor product rules can also be applied in the presence of trimming curves with an indicator function that matches curve visibility, although this direct approach would require an unreasonably high-order quadrature scheme to be accurate on all types of trimming curves. More sophisticated techniques have been developed to evaluate integrals over planar regions bound by curves [59, 29], which can be readily extended to represent the relevant domain of integration for a trimmed NURBS surface, ultimately placing quadrature nodes only on the visible portions of the surface [60]. However, even these more specialized quadrature schemes do not address the fundamental issue of the near-singular behavior of the integrand for query points near the surface.

6.4 Methods

In this work, we present a generalized winding number algorithm that can be accurately evaluated at arbitrary points in space, regardless of their distance to the ground truth CAD model, up to an arbitrary user tolerance. We assume that the surface is a trimmed tensor-product NURBS patch, with trimming curves that are explicitly defined in the parameter space of the knot spans of the patches by 2D NURBS curves. For each query point, we evaluate the GWN independently for each surface in the model and sum the results to obtain the final GWN.

Our algorithm is derived from Stokes' theorem, which reduces the dimension of Equation 6.5 to an integral over the boundary of each surface. For notational simplicity, we assume that the surface S is translated so that the query point q is at the origin.

In summary, we evaluate the GWN at query point $q = 0$ with respect to the surface S by

evaluating

$$w_S = \frac{1}{4\pi} \oint_{\partial S} \left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0 \right\rangle \cdot d\Gamma \tag{6.7}$$
$$+ \sum_{\left\{\substack{\text{intersections of} \\ S \text{ and } z\text{-axis}}\right\}} \begin{cases} 0.5 & \text{if } q \text{ is on the } \textit{positive} \text{ side of } S, \\ -0.5 & \text{if } q \text{ is on the } \textit{negative} \text{ side of } S. \end{cases}$$

The first term is our Stokes'-derived boundary formulation, which we evaluate using an error-controlled adaptive Gaussian quadrature scheme (See Section 6.7.0.4), and the second is a correction term that accounts for the jump discontinuity in the GWN field over the surface.

Importantly, the nature of this calculation for a given query point (and the associated computational cost) varies considerably according its relative position to the surface and its boundary edges, as shown in Figure 6.1. We present the full algorithm to compute the GWN of an arbitrary surface in Algorithm 8.

- **Far-field**: For points that are far from the surface, i.e., outside some bounding box encompassing the surface, the number of intersections is known to be zero, or can be *made* to be zero by rotating the surface. We therefore evaluate the GWN directly with only our boundary formulation.

- **Near-field**: For points that are near the surface, we still use the same boundary formulation, but must also perform a line-surface intersection test to identify the appropriate correction term (See Section 6.7.0.1).

- **Edge-cases**: For points which are near to a boundary edge, the necessary correction term cannot be reliably computed, and so we instead augment the surface with additional trimming curves to extract the problematic portion. The remaining surface can be evaluated as the near-field case, and the extracted portion is processed separately, likely as a far-field case relative to the original query point.

In principle, the appropriate correction term can be computed from the result of an intersection test between the surface and an *arbitrary* line by subsequently rotating the surface so that

the line is parallel to the $z-$axis. Furthermore, there is benefit to selecting a line for which it is straight-forward to identify (or preferably avoid altogether) certain types of edge-cases. As intersections with lines that are near-tangent to the surface are the most difficult to identify and resolve via the proposed algorithm, we heuristically select a line with a direction equal to an approximated average surface normal, which is computed for each surface as preprocessing. In cases where the symmetry of the surface results in a average normal with near-zero magnitude, we instead select the line randomly.

We also acknowledge at this point a similar boundary-oriented approach suggested in the preprint [109], in which the set of boundary curves for a 3D surface are projected onto a unit sphere around the query point, partitioning its the unit sphere into watertight regions. The GWN of the query point is then evaluated as the sum of the integer winding number of each region, weighted by its surface area, a calculation whose precision depends on a discretization of the boundary. Importantly, this method also involves performing a line-surface intersection test, but does so for *all* configurations of query point and surface, and not just those in the near-field. Furthermore, our proposed method is far less sensitive to the precision of the line-surface intersection test, making it considerably more reliable (See Section 6.7.0.1).

In the following sections, we justify our method mathematically by first providing an overview of Stokes' theorem, and then describing how its application to this problem naturally delineates each of the three above cases. Finally, we describe our strategy for evaluating the GWN field at points which are coincident to the surface.

6.4.1 Reformulation with Stokes' Theorem

In its most general form, Stokes' theorem states that if a vector field $F(x)$ is defined with continuous first derivatives on a region in $\mathbb{R}^3$ containing the surface S, then

$$\iint_S \nabla \times F \cdot dS = \oint_{\partial S} F \cdot d\Gamma, \tag{6.8}$$

where ∂S is the total boundary of the surface, which may contain multiple disconnected curves.

Evaluating such an integral via numerical quadrature along the lower-dimensional space naturally improves the efficiency of the total method. This is a well-established advantage of Stokes' theorem, and is indeed the basis of the method proposed by Gunderman et al. [60] for the purposes of evaluating integrals on trimmed NURBS surfaces. In cases where the "antiderivative" F can be evaluated exactly for a given definition of $\nabla \times F$, the number of quadrature nodes in a scheme of fixed order n is reduced from $O(n^2)$ points on the surface to $O(n)$ points on the boundary. This approach is less effective for arbitrary surface integrands, however, as computing F numerically at each quadrature node on the boundary typically returns the complexity of the quadrature scheme returns to $O(n^2)$.

In this problem context, the use of Stokes' theorem fundamentally improves the stability of the ensuing quadrature scheme, as query points which are near to the surface but not the boundary are not subject to the same near-singular behavior in the integrand. Furthermore, we capitalize on this boundary reformulation through our use of a geometrically adaptive quadrature method, the computational burden of which is dramatically reduced by considering subdivision only of 3D space curves.

To apply Stokes' theorem to the integral in Equation 6.5, we take

$$\nabla \times F = \frac{1}{4\pi} \cdot \frac{x}{||x||^3}. \tag{6.9}$$

Recall that we have made the assumption that the query point q is located at the origin, and so this integrand is continuous on the surface whenever $x \neq 0$, i.e., the query point is not coincident with the surface. (We treat this case explicitly in Section 6.4.2).

Given this definition of $\nabla \times F$, we are interested in its "antiderivative" F. There are many distinct vector fields that satisfy Equation 6.9, and we consider one such field in Cartesian coordinates:

$$F(x,y,z) = \frac{1}{4\pi} \left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0 \right\rangle. \tag{6.10}$$

If the theoretical conditions of Stokes' theorem can be met, then we can evaluate the GWN

for an arbitrary surface with the following 1D line integral,

$$w_S = \frac{1}{4\pi} \oint_{\partial S} \left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0 \right\rangle \cdot d\Gamma, \tag{6.11}$$

a reformulation of the problem statement that comes with a number of computational advantages.

However, the formulation of F in Equation 6.11 introduces some theoretical difficulties. For the conditions of Stokes' theorem to be met, both the original vector field and its antiderivative must be continuous not just on its domain of integration (the boundary of the surface) but also on the *original* domain of integration (the surface itself). Notably, this antiderivative is discontinuous whenever $x^2 + y^2 = 0$ on the surface, i.e., whenever the surface intersects an infinite line extending in both z directions from the query point. In such cases, we cannot mathematically justify the direct use of Stokes' theorem, even if the resulting line integral is itself continuous.

However, this particular antiderivative F, characterized by a line of discontinuities on the z-axis, was chosen in this section somewhat arbitrarily. Indeed, one can analogously define a *family* of antiderivatives F, each characterized by a *different* line of discontinuities. Computationally, it is practical to consider these different choices for F not as explicit vector fields, bur rather as a rotation of the surface so that the desired line of discontinuities is aligned to the z-axis. In this way, applying Stokes' theorem requires choosing an antiderivative F whose corresponding line of discontinuities does not intersect the surface, and rotating the surface accordingly.

In Figure 6.4, we see examples of different configurations of the surface and the line, and how the surface can be rotated to avoid the line of discontinuities. In practice, however, one cannot always identify such a line for query points which are nearer to the surface, nor is it necessarily the case that one exists (see Figure 6.4(c)).

In the following sections, we describe and justify the use of a correction term for our boundary formulation that directly accounts for the discontinuity in the antiderivative F, as well as a robust treatment of edge cases. This permits the use of the boundary formulation in Equation 6.11 for *all* possible query points, the computational benefits of which far outweigh the marginal costs of identifying the case and processing the surface as needed.

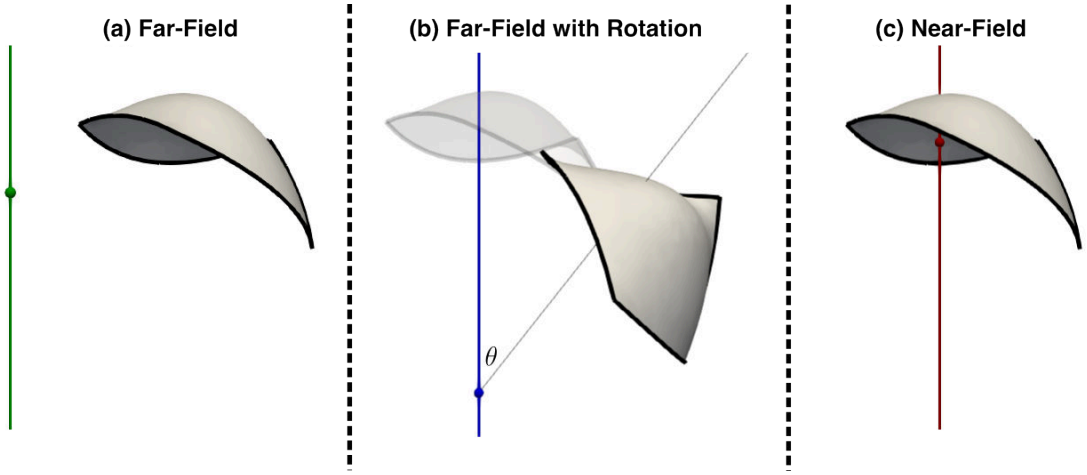


Figure 6.4: We classify far- and near-field points according to whether the selected line of discontinuities for antiderivative F either (a) does not intersect the surface, (b) can be rotated to avoid intersecting the surface, or (c) unavoidably intersects the surface.

6.4.1.1 Far-field: Direct Application of Stokes' Theorem

In cases where the query point is outside a bounding box of the surface, we can easily construct a line L which contains the query, but does not intersect the surface. This line corresponds to a particular form of the antiderivative F that is continuous on the surface, and so we can directly apply Stokes' theorem to evaluate the GWN. In addition to being the most straightforward application of Stokes' theorem in this work, is also the most common since an arbitrary query point will be considered far from all but a few surfaces in a typical CAD model.

Although there exists an antiderivative F that corresponds to any line L, it is somewhat tedious to construct F explicitly for an arbitrary line. Instead, we prefer to use Equation 6.11 during numerical evaluation, which assumes that the line of discontinuities is aligned to the z-axis. This means that in cases where L is not already aligned with the z-axis, we rotate the surface so that this is the case.

It may also be the case that a z-aligned line intersects the surface, but a line in the direction of the x- or y-axis does not. In these cases too there are simple analytic forms of antiderivative F, and so our implementation uses them directly in place of Equation 6.11, as this does not require

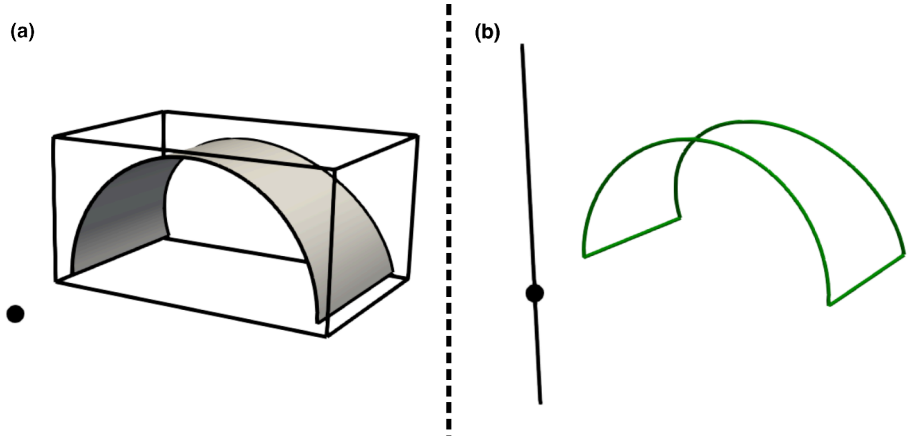


Figure 6.5: Far-Field Evaluation: For points that are exterior to a bounding box containing the surface (a), we directly apply Stokes' Theorem and evaluate the GWN as an integral along the surface's boundary curves (b).

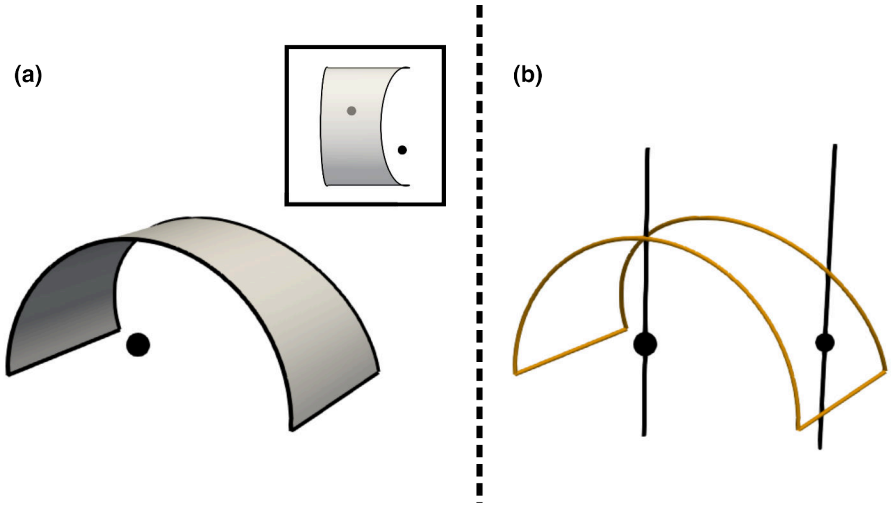


Figure 6.6: Near-Field Evaluation: For points that are interior to a bounding box, the Stokes' Theorem reformulation cannot be applied immediately. Instead, we can consider the points of intersection between the surface and a line containing the query. If an intersection occurs well within the interior of the surface, then we can proceed via integration of the same integrand in Equation 6.11 while accounting for each intersection (if any) by adding or subtracting a value of 0.5.

rotation of the surface:

$$w_{S,x\text{-axis}} = \frac{1}{4\pi} \oint_{\partial S} \left\langle \frac{zx}{(y^2+z^2)\,||\vec{x}||}, \frac{-yx}{(y^2+z^2)\,||\vec{x}||}, 0 \right\rangle \cdot d\Gamma \tag{6.12}$$

and

$$w_{S,y\text{-axis}} = \frac{1}{4\pi} \oint_{\partial S} \left\langle \frac{xy}{(x^2+z^2)\,||\vec{x}||}, \frac{-zy}{(x^2+z^2)\,||\vec{x}||}, 0 \right\rangle \cdot d\Gamma. \tag{6.13}$$

6.4.1.2 Near-field: Analytically Adjusted Stokes' Theorem

In cases where the query point is inside a bounding box of the surface, we can no longer guarantee that there exists a line L which does not intersect the surface. Instead, we consider an arbitrary direction for L, and must explicitly test for intersection between L and the surface. In this work, we take the simpler near-field case to be one where all intersections between L and the surface occur away from the surface edges. We also require each intersection to be non-degenerate, in the sense that L is not tangent to the surface, and the surface has non-zero normal at each point of intersection. When these conditions for each intersection are not met, the query point is handled as an edge-case.

Naturally, distinguishing the near-field and edge cases requires first identifying the location of all intersections between the surface and the line in the parameter space of the surface. As noted in Section 6.3, there are many numerical ray-casting algorithms one could apply to find intersections between this line and the surface, and in principle, any can be used for the purposes of evaluating the GWN. We describe our own geometric subdivision approach in Section 6.7.0.4, which in contrast to many algebraic alternatives, is designed to take advantage of the broader GWN algorithm's high tolerance for imprecision. It is this imprecision that ultimately makes this method considerably more robust than traditional ray-casting methods, as we are essentially able to perform a *different* surface-line intersection calculation for each surface in the model, with most surfaces not requiring this calculation at all.

Strictly speaking, any intersection between L and the surface violates the conditions of Stokes' theorem. However, we propose to compute the GWN using the same boundary integral formulation as Equation 6.11, but with the condition that for each intersection, the value 0.5 is added or subtracted according to the orientation of the surface at the point of intersection, resulting in Equation 6.19. Essentially, this adjustment compensates for the jump discontinuity present in the

GWN field along the surface.

We motivate the use of an explicit correction term by means of analogy in two spatial dimensions, and present a rigorous proof in Section 6.5. As summarized in Section 6.3.2, the GWN for an arbitrary curve in 2D can always be computed in terms of (1) an integer winding number for the curve closed by a straight line and (2) a generalized winding number defined by that straight line closure. An important observation, however, is that the fractional component of the curve's total GWN is derived exclusively from its boundary (the two points which define the curve's closure), and is otherwise completely independent of the curve's internal geometry. Indeed, given a function for the GWN field of the closure, in some sense the curve's internal geometry merely determines whether we add or subtract integer valued constants to the fractional value provided by this function to obtain the correct GWN of the curved shape. This can be seen in Figure 6.7, where the difference in the GWN field between the curve and its closure is integer valued.

Returning to 3D, one can also consider the evaluation of the GWN via Stokes' Theorem to always begin with a fractional value that is derived exclusively from the surface boundary, without any knowledge of the surface's internal geometry. With this value computed, one must compensate for the internal geometry via the addition of half-integer values determined through the number and orientation of intersections of a line with the surface (see Figure 6.8).

It is through this connection that we draw our closest analogy to the treatment of the 2D GWN in the previous chapter. In both cases, the GWN field is evaluated close to the shape with the same boundary formulation that defines the GWN in the far-field, but must adjust the value according to the shape's internal geometry. In 2D, the relevant boundary formulation is a simple subtended angle, while in 3D it is not the GWN of any particular shape, but rather the unaltered boundary integral in Equation 6.11.

6.4.1.3 Edge Cases: Disk Extraction via Trimming Curves

We now consider the remaining edge cases for which the query point is inside the surface bounding box, but the correction term in Equation 6.19 is theoretically unfounded. This occurs

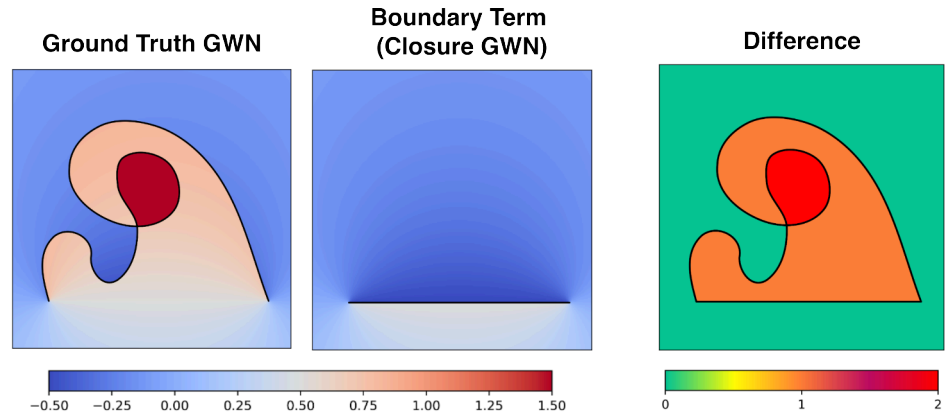


Figure 6.7: In 2D, the difference between the GWN for a curve (left) and the value of a particular boundary term (in this case the GWN of a linear closure) (middle) is always integer valued, and partitioned by the geometry (right). This suggests finding the GWN for the curve by evaluating the boundary term, and adjusting it using integer values determined by the curve.

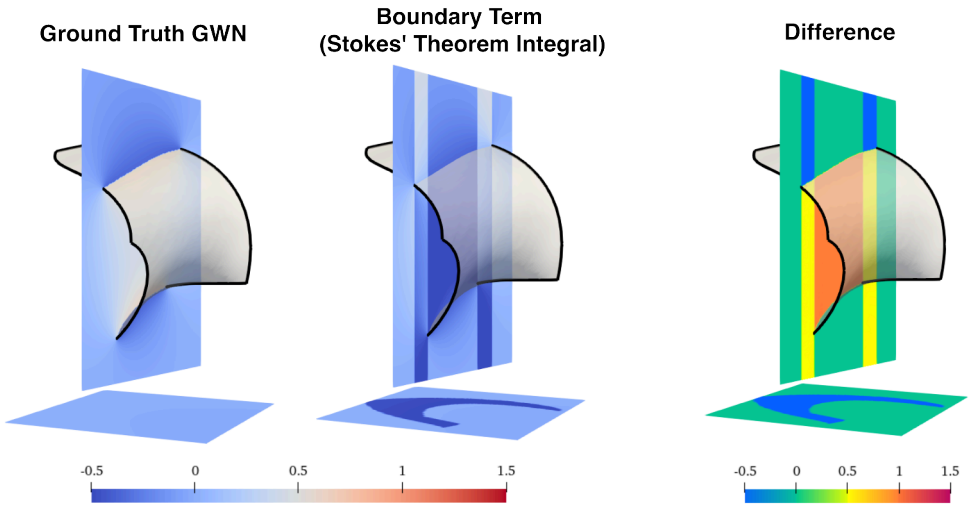


Figure 6.8: Analogous to the 2D case, the difference between the 3D GWN for a curved surface (left, shown at two planar slices) and the value of a particular boundary term (middle) is always *half*-integer valued, and partitioned by the geometry (right).

when the line of discontinuities intersects the surface at a boundary, at a cusp, or is tangent to the surface (See Figure 6.9). We treat all edge cases uniformly, by removing from the surface a disk in parameter space around the point of intersection. This guaratnees that the line does not intersect the surface at the otherwise problematic intersection point, and permits evaluation via Equation 6.19. The extracted disk is then fed back into the original algorithm for re-processing.

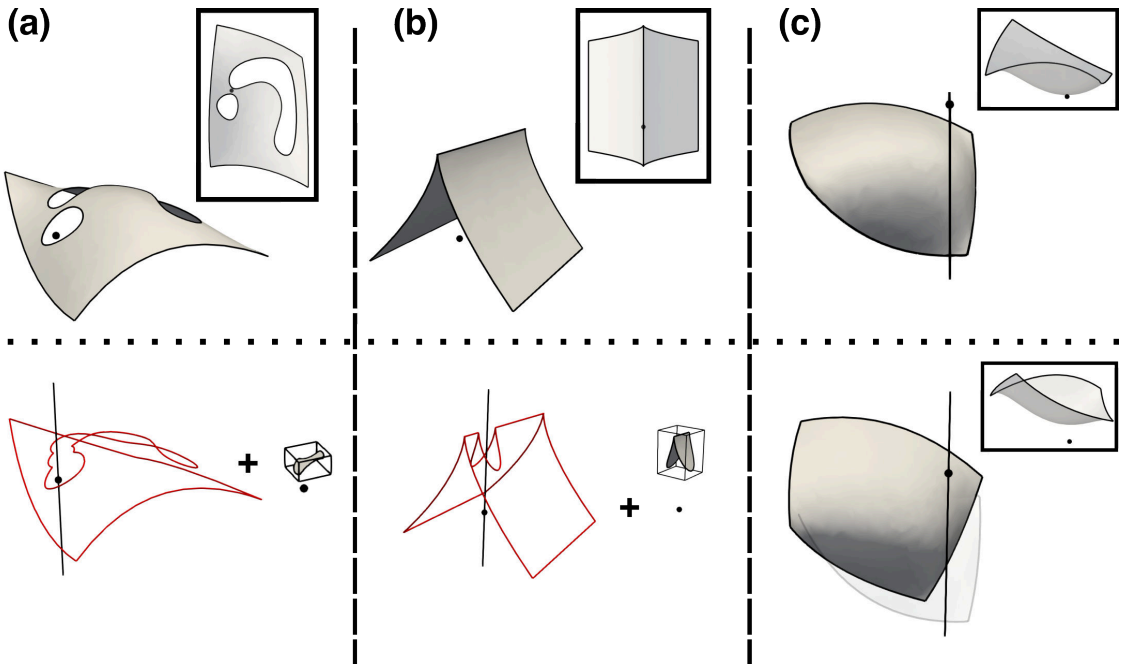


Figure 6.9: Edge-Case Evaluation: For all points that are interior to a bounding box, we consider a line containing the query point. If the line intersects the surface at (a) a point on the surface near to the boundary or (b) a point for which a normal cannot be defined (i.e., a cusp), we clip the surface along a small disk in parameter space. This removes the intersection with the original surface, and the remaining disk can be fed back into our algorithm for reprocessing. If the intersection occurs at (c) a tangent point between the line and the surface, we apply a random rotation to the surface that does not change the GWN evaluation, but is likely to result in a case captured by the procedure in Section 6.4.1.2.

Importantly, the produced disk occupies a much smaller bounding box than the original surface, and so the computational cost of the re-processing is significantly reduced (See Figure 6.19).

The choice of disk radius is mathematically arbitrary, but we have found empirically that fixing the radius r to be 1% of the bounding box diagonal is sufficient to ensure that the disk is small enough to be processed efficiently, but large enough to avoid numerical instability in later quadrature. The more important consideration, however, is that the disk is large enough to compensate for any numerical imprecision in the intersection calculation, in the sense that a disk centered at the *recorded* intersection point will also contain the *actual* intersection point (See Section 6.7.0.1). Our strategy for handling edge cases is such that it could be used to handle *any* query point at increased computional cost, and so we also conservatively treat any intersection

within the same distance r of the surface boundary as an edge case. Altogether, this ensures that the accuracy of the overall GWN evaluation is robust to the intersection calculation.

The only edge case which is not well-handled by this strategy is the case of a line that is tangent to the surface, as the remaining surface may still intersect the line. In these cases, we instead *rotate* the surface (i.e., consider a different line) and feed the rotated surface into the original algorithm (See Figure 6.9(c)). While this is the most computationally expensive edge case to handle, it is also the most uncommon.

6.4.2 GWN for Coincident Points

Strictly speaking, the generalized winding number of query points that are coincident with the surface is undefined, as the fundamental mathematical definition of the GWN, (Equation 6.9) is discontinuous. On a conceptual level, we require that our algorithm maintain exact geometric fidelity for completely arbitrary query points, which includes those coincident with the surface. However, there are many practical reasons why our algorithm should always return a sensible numeric value for the GWN that is mathematically justified according to some convention.

First and foremost, returning reasonable values for coincident winding numbers makes the method more amenable to downstream applications of the GWN field, many of which will expect some intuitive value to be returned. For example, if the shape is indeed watertight, then the GWN of all coincident points should be half-integer valued. This is a largely intuitive choice, as it means that when winding numbers are mapped to a containment decision via a simple strategy like rounding, all coincident points are treated equivalently.

Winding numbers are typically computed in batches without any prior knowledge as to their position relative to the surface, but it is not difficult to imagine a circumstance in which large, structured groups of query points do indeed coincide with some similarly structured portion of the surface. In this case, it is important that the GWN results in numerical values that are consistent with one another and the rest of the field. This prohibits the common solution of simply perturbing the query points slightly to avoid coincidence, as this may move truly interior or exterior points

Algorithm 8: `TrimmedSurfaceGWN` Evaluate the generalized winding number for a trimmed NURBS surface.

Input: S: Trimmed NURBS surface
q: Query point
Output: w_S: The GWN evaluated at q

$\Gamma \leftarrow \{\}$ `// Initialize the 1D curves to later integrate over`
$w_S = 0$ `// Initialize the GWN`
$S \leftarrow S - q$ `// Shift the patch so the query point is at origin`

if $q \notin$ `BoundingBox`(S) **then**
  `/* Rotate patch so` z`-axis does not intersect patch */`
  $L \leftarrow$ `Non-intersecting Line`
  $S \leftarrow$ `RotateToZ`(S, L)
  $\Gamma \leftarrow$ `BoundaryCurves`(S)
else
  `/* Compute approximate intersections between` z`-axis and` S `*/`
  $z, u, v \leftarrow$ `LinePatchIntersection`$(S, \text{z-axis}, \epsilon_{ls})$
  foreach intersection point $\{(u_0, v_0), z_0\}$ **do**
    $\vec{n} \leftarrow S_u(u_0, v_0) \times S_v(u_0, v_0)$
    if $S(u_0, v_0)$ **is an interior point of** S **with normal** $\vec{n} \neq 0$ **then**
      `/* Add a constant to the GWN according to orientation */`
      if $\vec{n}_z z_0 > 0$ **then**
        $w_S = w_S - 0.5$
      else if $\vec{n}_z z_0 < 0$ **then**
        $w_S = w_S + 0.5$
      else
        $w_S = w_S + 0.0$
    else if S **is tangent to the** z**-axis at** $S(u_0, v_0)$ **then**
      `/* Rotate the surface and try again */`
      return `TrimmedSurfaceGWN(RandomRotation`$(S), 0)$
    else
      `/* Extract a disk in parameter space from S with radius equal to 1% of the patch parameter space bounding box */`
      $S, S_{\text{disk}} \leftarrow$ `ExtractParameterDisk`(S, D)
      $w_S = w_S +$ `TrimmedSurfaceGWN`$(S_{\text{disk}}, 0)$
      `/* Proceed with the rest of the surface */`
  $\Gamma \leftarrow$ `BoundaryCurves`(S)
foreach 1D Curve in Γ **do**
  $w_S = w_S +$ `EvaluateLineIntegral`(C, ϵ_q)
return w_S

across a boundary.

This is a problem unique to curved shapes, as the GWN of a point coincident with a line or triangle is straightforward to identify and exactly set equal to zero. We address the analogous issue for 2D curves in the previous chapter by defining the winding number of coincident points to be the average of the value across the jump discontinuity of the scalar field. This value can be computed exactly in 2D, as it is equivalent to the value of the angle subtended by a tangent line to the curve at the query point and each endpoint. The difficulty then is is identifying when the query point is coincident with the curve, as doing so directly is unnecessary for all but coincident points.

In 3D, we analogously take the winding number of coincident points to be the average of the values computed across the jump discontinuity of the scalar field. However, the line-surface intersection routine we call for near-field evaluation of the GWN field immediately identifies such points, which allows for more explicit handling of the discontinuity. For example, if the intersection point is known to be interior within the surface, then the jump discontinuity is known by Equation 6.19 to contribute a value of 0.5 on one side and -0.5 on the other. This means that if the origin is *at* the intersection, we add the average of these two values, i.e., zero, to the integral evaluated along the boundary curves.

The only unaddressed case is when the query point is located coincident with a boundary curve of the surface. This case is uniquely difficult to solve via our Stokes' theorem reformulation, as the evaluated *antiderivative* is non-trivially discontinuous along its domain. Similarly, this prevents the use of more general quadrature strategies that are built around dimension reduction via Stokes' Theorem such as that described by Gunderman et al. [60]. We handle these edge cases in much the same way as in Section 6.4.1.3, where we remove a disk from the surface around the query point. The GWN calculation for the larger subdivided surface is performed as normal, but we consider the GWN of the removed disk to be zero, an assumption that only holds in the near-field *because* the query point is coincident to the disk. To strengthen this assumption, we remove a much smaller disk than in the general case.

Although this procedure does introduce some small amount of error in the overall calculation, we find it to be largely insignificant for downstream containment queries, as the side to which a given half-integer GWN value is rounded is largely dependent on the floating point error throughout the broader method. However, one could address such points more directly with other integration techniques. Of particular note is that, the analogous discontinuity in the *original* surface integrand in Equation 6.5 is, in fact, removable in the case of coincident points. Somewhat paradoxically, this means that evaluating the surface integral for coincident points is considerably more stable than would otherwise be the case for near-field query points. Therefore we handle this case with a more standard technique that avoids placing quadrature nodes on the boundary altogether, such as the method of Chin et al. [29] to perform the necessary integral directly over the portion of the patches' parameter space that is made visible by the trimming curves. Naturally this 2D surface integral would be considerably more computationally expensive than our Stokes' theorem formulation, but it would only need to be applied for a small subset of query points to achieve accurate results.

6.5 Analytic discontinuity fix for Near-Field GWN

We now provide a formal proof of the correction term in Equation 6.19, which suggests that one can account for discontinuities in the Stokes' theorem formulation analytically by adding or subtracting 0.5 for each intersection between the surface and the z-axis.

As in Section 6.4.1.2, we assume that all intersections occur on the strict interior of the surface, have well-defined normal vectors on the surface at the point of intersection (i.e., do not occur on cusps of the surface), and are such that the z-axis is not tangent to the surface. An important consequence of these assumptions is that there is necessarily a neighborhood around the point of intersection which is locally flat, meaning there is some other line in 3D space that does not intersect the surface within this neighborhood. As established in Section 6.4.1.1, this means that the value of the winding number with respect to the portion of the surface within this neighborhood depends only on the neighborhood's boundary. In other words, one can deform the surface in the interior of this neighborhood without changing the winding number evaluated over

the entire surface.

In particular, we consider a deformation of the surface within this neighborhood so that within it there exists a flat disk that is orthogonal to the z-axis, a process akin to to "flattening" the surface around points of intersection between the z-axis and the surface. Importantly, this procedure does not alter the value of the generalized winding number, nor its numerical calculation within the proposed algorithm, but does considerably simplify its analytic evaluation within subsequent steps of this proof.

To reiterate, our objective is to evaluate the generalized winding number with respect to the surface S by computing numerically only an integral along its boundary. Thus far, we have demonstrated that this task is equivalent to evaluating the generalized winding number with respect to a surface which is deformed around points of intersection with the z-axis. For notational simplicity, we continue to refer to this deformed surface as S. To evaluate w_S, we consider the relevant surface integral to be partitioned by the flat disk, D, which is constructed by our choice of deformation, such that

$$\begin{aligned} w_S &= \frac{1}{4\pi} \iint_S \frac{\vec{x} \cdot \hat{n}}{||\vec{x}||^3} \, dS, \\ &= \frac{1}{4\pi} \iint_D \frac{\vec{x} \cdot \hat{n}}{||\vec{x}||^3} \, dS + \frac{1}{4\pi} \iint_{S-D} \frac{\vec{x} \cdot \hat{n}}{||\vec{x}||^3} \, dS. \end{aligned}$$

We assume at this point, without loss of generality, that there is only a single point of intersection between the surface and the z-axis, recognizing that each point of intersection results in an additional disk, and an additional partition of the integral which can be evaluated in the same way.

By construction, the punctured surface $S - D$ does not intersect the z-axis, and so we can

apply Stokes' Theorem to this portion of the integral:

$$\begin{aligned}
w_S &= \frac{1}{4\pi}\iint_D \frac{\vec{x}\cdot\hat{n}}{||\vec{x}||^3}\,dS + \frac{1}{4\pi}\iint_{S-D} \frac{\vec{x}\cdot\hat{n}}{||\vec{x}||^3}\,dS \\
&= \frac{1}{4\pi}\iint_D \frac{\vec{x}\cdot\hat{n}}{||\vec{x}||^3}\,dS \\
&+ \frac{1}{4\pi}\oint_{\partial(S-D)} \left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0 \right\rangle \cdot d\Gamma \\
&= \frac{1}{4\pi}\iint_D \frac{\vec{x}\cdot\hat{n}}{||\vec{x}||^3}\,dS \\
&+ \frac{1}{4\pi}\oint_{\partial D^c} \left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0 \right\rangle \cdot d\Gamma \\
&+ \frac{1}{4\pi}\oint_{\partial S} \left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0 \right\rangle \cdot d\Gamma,
\end{aligned}$$

where ∂S is the total boundary of S *prior* to the removal of the disk D, and ∂D^c is the boundary of the disk, oriented *negatively* so that its orientation is consistent with ∂S. Of these three terms, we only compute numerically the integral of the line around the boundary ∂S, as described in Section 6.7, i.e., the same collection of curves over which we integrate to find w_S in the case where S does not intersect the z-axis. For the remaining terms, we instead consider their behavior as the radius of the disk D approaches zero.

The more direct of the two is the surface integral over the disk D. Because this value is exactly equal to w_D, i.e., the winding number of the surface D with respect to the origin, it necessarily goes to 0 as the disk decreases in surface area. Therefore it remains to show that the line integral around ∂D^c approaches ± 0.5 as the radius of the disk approaches 0.

By the construction of the deformed surface, we can parameterize the boundary of this disk on $t \in [0,1]$ straightforwardly as

$$\Gamma(t) = \langle r\cos(2\pi t), -r\sin(2\pi t), z_0 \rangle\,,$$

where $z = z_0 \neq 0$ is the center of the disk and r is its radius. We substitute this parameterization of $\Gamma(t)$ and the curve differential $d\Gamma$ into the integral in question and after some simplification, we have

$$\frac{1}{4\pi}\oint_{\partial D^c}\left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0\right\rangle \cdot d\Gamma \tag{6.14}$$

$$= \frac{1}{4\pi}\int_0^1 \frac{-r\sin(2\pi t)\cdot z_0 \cdot -2\pi r\sin(2\pi t) - r\cos(2\pi t)\cdot z_0 \cdot -2\pi r\cos(2\pi t)}{(r^2\cos^2(2\pi t)+r^2\sin^2(2\pi t))\sqrt{r^2\cos^2(2\pi t)+r^2\sin^2(2\pi t)+z_0^2}}\,dt \tag{6.15}$$

$$= \frac{1}{2}\int_0^1 \frac{z_0 r^2(\sin^2(2\pi t)+\cos^2(2\pi t))}{r^2\sqrt{r^2+z_0^2}}\,dt \tag{6.16}$$

$$= -\frac{z_0}{2}\int_0^1 \frac{1}{\sqrt{r^2+z_0^2}}\,dt \tag{6.17}$$

$$= -\frac{z_0}{2\sqrt{r^2+z_0^2}}. \tag{6.18}$$

Clearly, as $r \to 0^+$, we have that this value approaches -0.5 if $z_0 > 0$ and 0.5 if $z_0 < 0$. Altogether, in the case where there is an intersection between between the z-axis and the *original* surface S, we can write

$$w_S = \frac{1}{4\pi}\oint_{\partial S}\left\langle \frac{yz}{(x^2+y^2)\,||\vec{x}||}, \frac{-xz}{(x^2+y^2)\,||\vec{x}||}, 0\right\rangle \cdot d\Gamma + \begin{cases} 0.5 & \text{if the origin is } \textit{above} \text{ the intersection point,} \\ -0.5 & \text{if the origin is } \textit{below} \text{ the intersection point.} \end{cases}$$

In essence, this means that in many cases where Stokes' Theorem does not otherwise apply due to the presence points on the surface that intersect with the z-axis, we can practically compute the generalized winding number by evaluating an integral along the boundary of the surface, and adding or subtracting the value 0.5 according to the orientation of the surface at this intersection.

6.6 Numerical Experiments and Results

6.6.1 Generalized Winding Numbers on Open CAD Models

We first justify our choice of a boundary formulation in our solution method by comparing the accuracy of a 2D numerical integration scheme that solves Equation 6.5 to a 1D numerical integration scheme that solves Equation 6.19. We perform this comparison by evaluating the GWN field at two slices of the sphere model from Figure 6.11(d) using a simple and direct application of

a 20-point Gaussian quadrature rule. The derived 1D method involves applying this 20-point rule to each boundary curve for a total of $4 \cdot 20$ evaluations of the NURBS surface. On the other hand, the 2D scheme involves a tensor product 20-point rules for a total of 20^2 evaluations of the NURBS surface.

We show the results of this comparison in Figure 6.10, where we see that errors in the 2D scheme are present for all points near the surface, where errors in the 1D scheme are only present for points near the surface boundary. While it is true that both types of errors can be mitigated through the use of adaptive quadrature, applying such a strategy to a 2D scheme would quickly require unreasonably many surface evaluations to achieve the same level of accuracy as our adaptive 1D scheme. Not to mention the fact that the accuracy of the 2D scheme is further compromised by the presence of trimming curves, which are far more naturally handled in the 1D scheme.

We now turn our attention to the full implmenetation of our algorithm, which solves Equation 6.19 at arbitrary points up to arbitrary accuracy with a more sophisticated, error-controlled adaptive quadrature scheme (See Section 6.7.0.4). To demonstrate the versatility of our algorithm, we evaluate the GWN on a number of example CAD models taken from the ABC dataset [91] (see Figure 6.11). For the shapes in Figure 6.11(a-c), we have intentionally and arbitrarily made the models non-watertight by removing portions of the surface for the purposes of demonstration.

For both watertight shapes and the portions of non-watertight shapes that are far from disconnected boundaries, we observe that the GWN field neatly partitions the domain into exterior and interior regions. In this way, it is clear for values which are near integers as to how the corresponding point should be classified: points where the GWN field is close to 0 where the query point should be considered outside the shape, and close to 1 where the query point should be considered inside the shape.

Closer to these gaps in the shape, however, we see that the GWN field has degraded to values with a larger fractional component. In some sense, this degradation depicts an increasing amount of uncertainty in the subsequent containment classification. The simplest strategy to map these fractional values to a containment decision is to round them to the nearest integer, as this

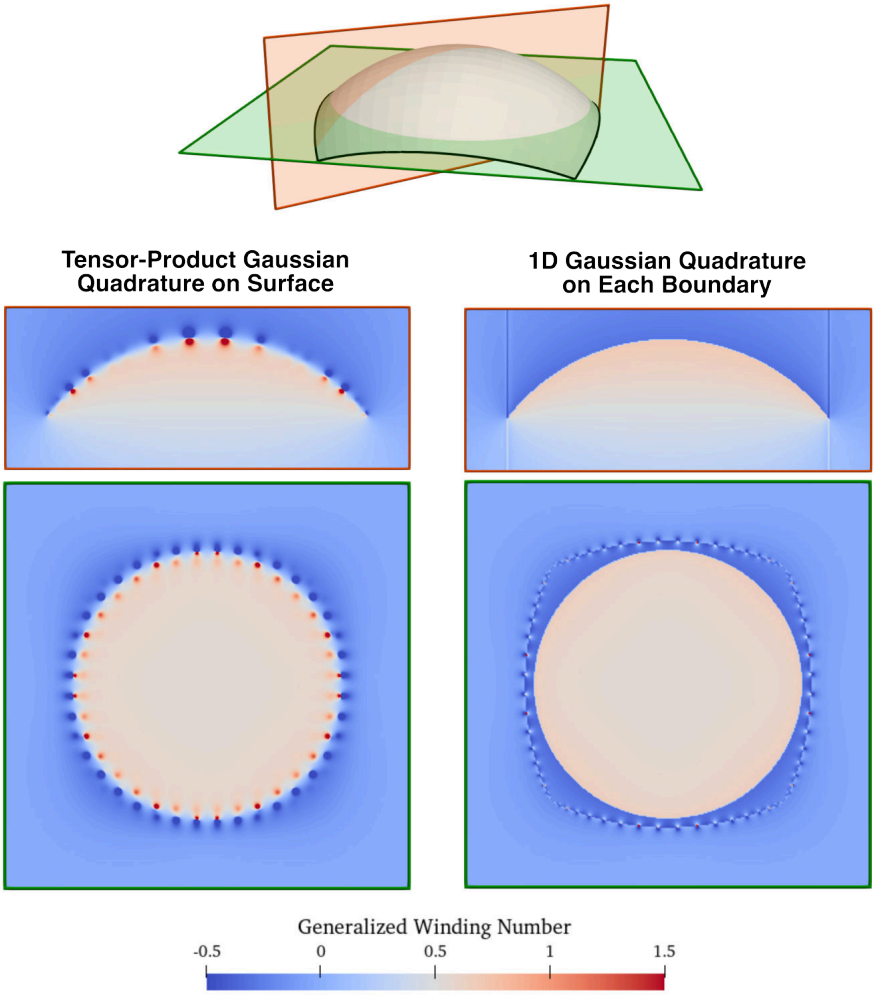


Figure 6.10: We compare equivalent *fixed-order, non-adaptive* numerical integration techniques to solve (a) Equation 6.5 via a 2D tensor product Gaussian quadrature scheme and (b) Equation 6.19 via a 1D Gaussian quadrature scheme. Although both methods utilize the same 20-point Gaussian quadrature rule, the derived 2D method requires evaluating the surface 20^2 times, while the derived 1D method requires doing so $4 \cdot 20$ times. Nevertheless, the scheme in (b) results in a more sparse distribution of numerical errors.

is guaranteed to agree with the boundary for points that are nearby and provides a reasonable classification for points which are further away.

As examples, we consider in Figure 6.12 the non-watertight shapes in Figure 6.11(a, c), and construct a 0.5-isosurface of the GWN field. In the case of the "spring" shape, the isosurface does indeed faithfully represent a flat boundary at the end of the coil. On the other hand, a nearly identical gap in the "bobbin" shape has a corresponding isosurface with a concave dent. This can

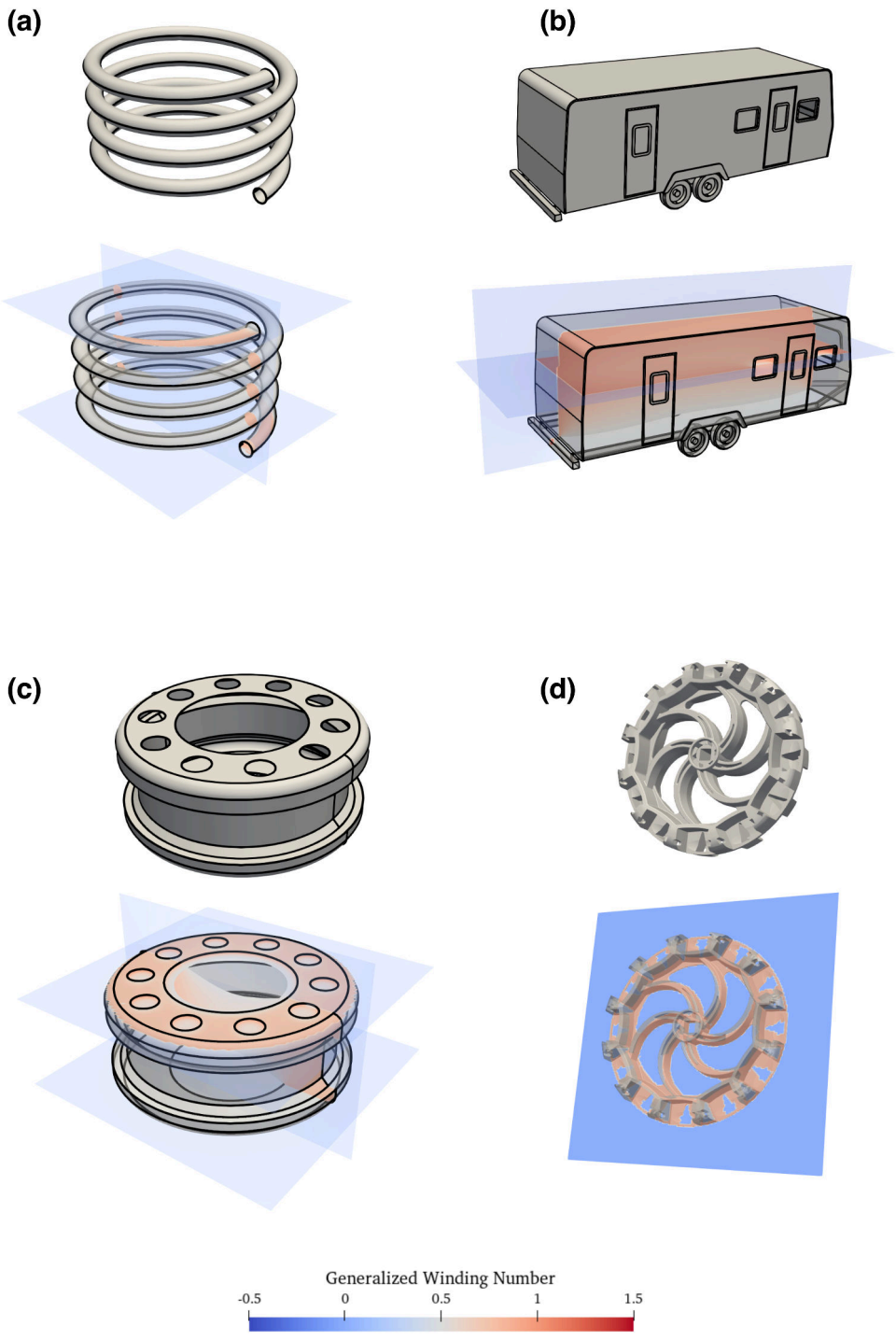


Figure 6.11: We demonstrate our calculation of the GWN field on shapes of varying size, complexity, and watertightness. The 3D scalar field is viewed on various non-planar slices through the model where, for clarity of illustration, we have increased the opacity of all values less than 0.5, i.e., points which would be classified as "exterior" by a simple rounding heuristic.

be attributed to the fact that the GWN is a *global* property of the bounding geometry. Despite the GWN field rapidly decaying away from each surface in the model, far-away parts of the shape nevertheless carry a non-negligible influence on the GWN field everywhere in space. In the "bobbin" shape (Figure 6.11(c)), the GWN at the points of the removed disk are, in fact, slightly less than 0.5 due to the presence of asymmetric gaps in the top and bottom of the shape. In contrast, the GWN in the "spring" shape (Figure 6.11(a)) is exactly 0.5 due to the gaps being coplanar with one another, which is largely a coincidence of the shape's specific geometry.

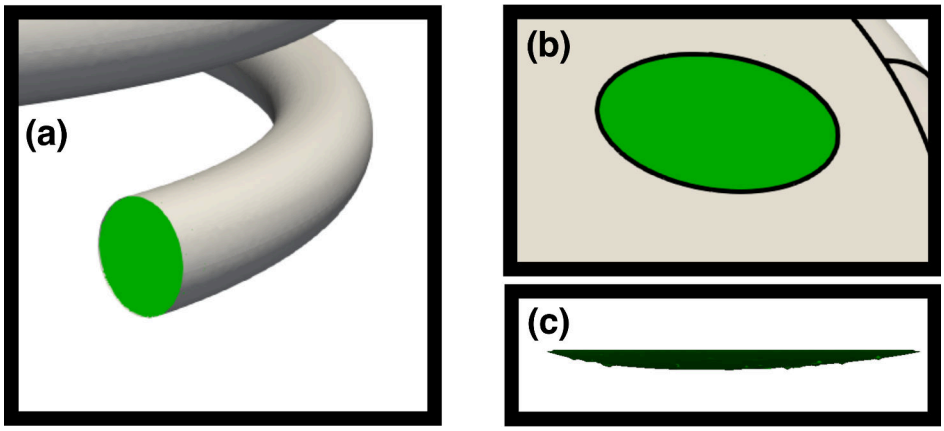


Figure 6.12: The fractional values of the GWN field can be mapped to a specific containment classification by rounding to the nearest integer. (a) For some shapes, this implied boundary between interior and exterior regions closely matches originally missing surfaces. (b) For many other surfaces, the implied boundary isosurface may be unexpectedly curved. (c) The surface in (b) viewed from the side.

More generally, we make no claim that the raw GWN field is an optimal proxy for the containment query in the much broader context of surface repair (see [48] for a thoughtful discussion of the topic). In this work however, we are primarily interested in applications of the GWN in contexts where the priority is faithfully representing input geometry, as opposed to describing an unknown, ground-truth surface from which the input geometry is a subset. Even still, the GWN field has been shown to be useful as *input* for other techniques, such as those that use more sophisticated classification algorithms [79] or use the GWN as a prior for more rigorous statistical analyses [151]. While we do not explore such applications further in this work, we believe that the accurate and efficient evaluation of the GWN field as presented is a necessary precursor to enabling

such techniques on more general CAD models.

6.6.2 Accuracy Evaluation

The key feature of Algorithm 8 is that it is capable of evaluating the GWN field exactly with respect to arbitrary input geometry. We can see this in Figure 6.13 by considering the GWN on a small, highly detailed subset of the complex gear model from Figure 6.11(d). In specific contrast with methods involving linear discretization into a triangle mesh or related format, this evaluation can be done at arbitrary points in space without the need for a potentially unknown amount of preprocessing. While modern surface tessellation schemes can use a linear deflection parameter that is sufficient for controlling how far the triangulation can be from the original shape, it can be difficult to know *a priori* if this level of discretization is necessary for the same level of accuracy at a particular point. This can lead to unnecessary computational expense in the form of globally dense triangulations. In contrast, by operating directly on the trimmed surfaces in a shape, our algorithm is adaptive for each query to the requested level of accuracy.

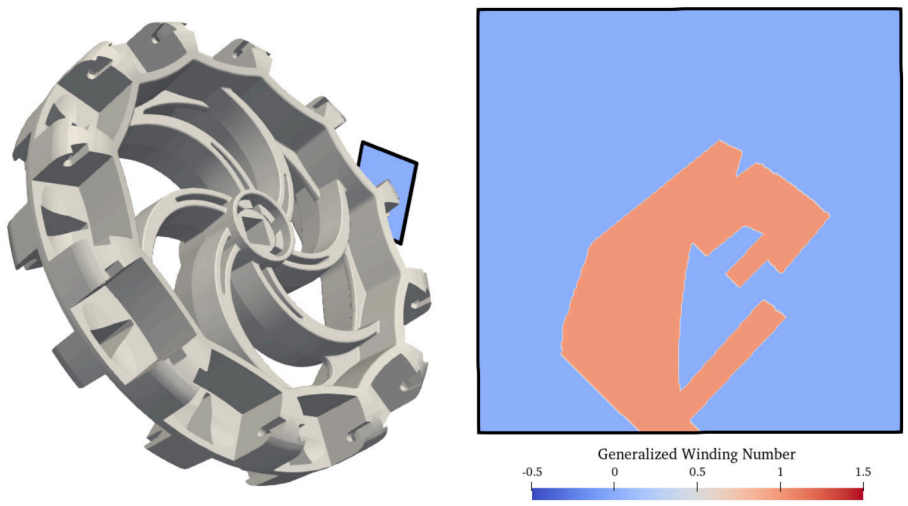


Figure 6.13: By operating directly on the trimmed surfaces of the CAD geometry, we are able to evaluate the GWN field exactly at arbitrary distances to the surface without additional preprocessing.

To see more quantifiably the effect of discretization on evaluation of the GWN field, we

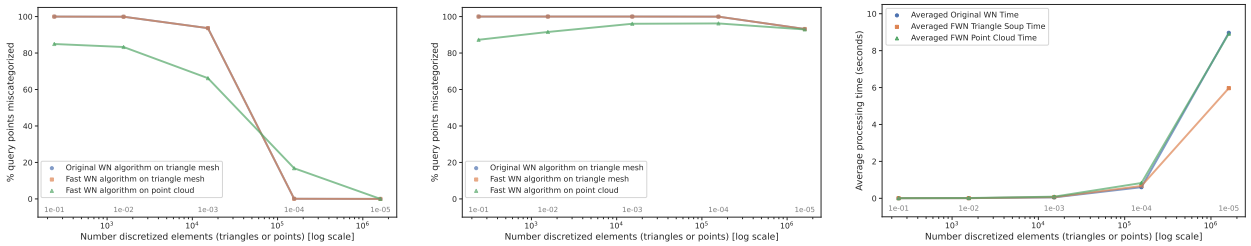


Figure 6.14: Linear discretizations of a NURBS unit sphere as triangle meshes and point clouds necessarily approximate the geometry. When applying existing linear WN approaches [79, 10] to these meshes, this leads to miscategorized containment queries for query points close to the surface, especially with lower resolution meshes. The charts show the percentage of 10k query points that are miscategorized by the different approximated approaches when the samples are taken at distances in the vicinity of 10^{-4} (left) and 10^{-6} (center) from the surface. The meshes are generated from the CAD model using Open Cascade's deflection parameter (gray annotations), corresponding to allowable distance from the discretized mesh to the CAD surface. At the same time, increased discretization resolutions lead to increased memory requirements (shown on the x-axis in terms of the number of discretized triangles or point primitives) and to higher averaged processing times (right plot). In our use of the point cloud method [10], we project the point samples to the surface of the sphere, and use the sphere normals. In all examples, we note that our GWN algorithm returns no misclassifications among all sampled points.

adapt existing GWN techniques for triangle meshes and point clouds to a non-watertight CAD model that has been tessellated at varying levels of **deflection** [124], i.e. sufficiently resolved triangulations to capture features up to a given distance from the surface. For the experiments in Figure 6.14, we utilized the libigl [80] implementations of the technique of [79] as applied to such a triangulated shape, which we refer to as "Original WN", as well as the libigl implementation of the algorithms from [10] for point clouds ("Fast WN on point cloud") and triangle meshes ("Fast WN triangle mesh"), computing an exact GWN field for the approximated geometry. Specifically, we used Open Cascade to generate several triangle meshes at specified deflection values ranging from 10^{-1} to 10^{-5} for a unit sphere NURBS model defined by six biquartic patches [33]. Each lower order of magnitude of deflection is satisfied by approximately an order of magnitude more triangles, with the highest resolution mesh requiring about 1.6 million triangles to resolve features at distances of at most 10^{-5} from the surface. We also generated a point cloud from each triangle mesh by taking a random sample within each corresponding triangle and projecting its position and normal to lie on the unit sphere. We then queried the GWN on these meshes using collections

of query points at successively closer distances to the surface. Figure 6.14(a) shows the percentage of ten thousand query points inside the sphere at a distance of 10^{-4} from the surface that were misclassified as outside the sphere, while Figure 6.14(b) shows the same results for a collection of points at a distance of around 10^{-6} from the surface. As can be seen from the figure, surfaces that were discretized using a deflection of 10^{-5} are able to properly resolve the containment of points at distances of 10^{-5}, but the coarser meshes cannot. While finer meshes are able to resolve closer points, the disretization grows exponentially larger, requiring more memory and processing time, as shown in Figure 6.14(c). In contrast, our method is able to accurately classify the entire collection of points without the need for an expensive intermediate discretization.

6.6.3 Performance Evaluation

We now demonstrate the performance of Algorithm 8. The following numerical experiments were conducted on an AMD Ryzen 7 5700 CPU with 32GB of RAM. These results are computed sequentially, although we note that the evaluation of the GWN field at many points is a highly parallelizable task.

Although a triangulated CAD model is composed of significantly more primitive components than the original collection of surfaces, methods that operate on CAD geometry are typically more expensive than equivalent operations on STL data types. However, our use of precomputed subdivision surfaces and efficient caching of per-curve quadrature nodes allows us to evaluate the GWN field at a practical rate, even without the use of more sophisticated hierarchical data structures, the implementation of which we largely consider to be future work.

In Table 6.1 we show the average time needed to evaluate each query point on a uniform grid of $50 \times 50 \times 50$ points located in a bounding box of each shape, as well as a per-query average. Across all query points, we also consider the average time needed to evaluate the GWN for each surface in the shape, further sorted between each of the three cases described in Section 6.4.1, i.e., far-field, near-field, and edge cases. We also record the fraction of all point-surface GWN calculations across the shape which are classified as each of these three cases to better understand the distribution of

computational cost.

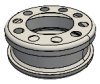
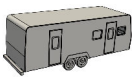
Shape	(a) 	(b) 	(c) 	(d) 
Number of Patches	2	18	228	1784
Number of Trimming Curves	8	76	1092	10256
% Far-field Evaluation	11.24%	82.73%	99.85%	98.96%
% Near-field Evaluation	45.37%	9.210%	0.107%	00.96%
% Edge Case Evaluation	43.38%	8.053%	0.038%	00.07%
Avg. Time across Query Points	0.0299	0.0054	0.00198	0.0431
Avg. Time for Far-field	1.125×10^{-5}	9.644×10^{-5}	0.000837	0.0083
Avg. Time for Near-field	0.000155	0.00143	0.000651	0.0303
Avg. Time for Edge Cases	0.0297	0.00385	0.000495	0.0044

Table 6.1: All times in seconds. We evaluate the GWN field at a uniform grid of query points for each shape, and record the average time spent to resolve each query. We also show the number of query points that are treated by each of the three cases described in Section 6.4.1, and the total time needed to resolve each of these cases. We further explore the performance of the spring model in Table 6.16.

From Table 6.1, we can see several key features of the performance of our algorithm. Most notably, we see that although the total time needed to evaluate the GWN field in principle scales linearly with the number of trimming curves, the total time is influenced far more by the distribution of query points into the three described cases. For example, although shape (c) has many more trimming curves than shape (b), it takes less time overall to evaluate its GWN at the selected query points, as a greater fraction of them are classified as far-field cases.

However, these results also demonstrate that the largest influence on run-time performance is the adaptive quadrature method used to guarantee accurate numerical results, and which itself is determined by the applicability of our memoization strategy as described in Section 6.7.0.4. This process caches the position of quadrature nodes along each trimming curve in 3D space as they are computed at each level of refinement. Such a strategy is highly effective at amortizing the cost of repeated evaluations of the GWN field for the same surface, but is less effective in cases like shapes (c) and (d), where the larger number of surfaces in the shapes means more query points would be needed to reach the same level of numerical efficiency. Furthermore, this memoization strategy cannot be straightforwardly applied to edge cases, as the process of disk extraction alters

the trimming curves, and therefore the placement of quadrature nodes.

As an example, shape (a) represents a perhaps unexpectedly challenging case for our algorithm where, despite being composed of only two untrimmed patches, it takes the longest to process. Because the two patches largely share a bounding box, there are few query points are classified as far-field from either, and of those that remain, many must be classified as edge cases. At the same time, these edge cases must be resolved without the computational benefits of memoization, and the long boundaries of each patch means that many levels of subdivision must be needed to accurately evaluate the GWN field at these points.

One strategy to compensate for this is to refine the surface, splitting it into smaller patches prior to evaluation. In contrast to the refinement of linear meshes, this operation does not change the numerical result of our algorithm, but can have a significant effect on performance. Indeed, by splitting the two patches of shape (a) according to their knot spans (i.e., Bézier extraction), we can improve computational performance by an order of magnitude, as shown in Figure 6.15 and Table 6.16. In this example, refinement causes many more query points to be classified as far-field, which can be evaluated with only the efficient cached quadrature.

6.7 Discussion

In this section, we highlight the computational components that support our implementation and numerical experiments. Although there are some specifications within these methods that are unique to our application, we expect that an alternate approach that meets these same requirements would be similarly suitable.

6.7.0.1 Line-Surface Intersections

To accurately evaluate the sum in Equation 6.19 requires a robust intersection test between an arbitrary line and an input trimmed NURBS surface. We reiterate that this test is only necessary in the near-field of the surface, or else the line can be selected according to the surface's bounding box to ensure no intersections could be recorded.

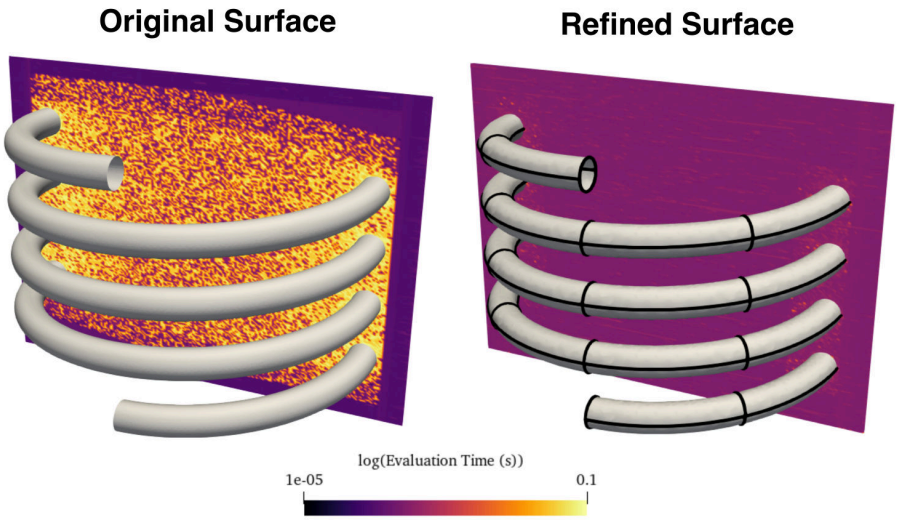


Figure 6.15: By subdividing the surface, we have considerably improved the performance of our algorithm. Although we have added more patches and therefore the number of integrals that must be evaluated, many more points are considered to be in the far-field of other patches, which permits use of more efficient quadrature rules.

Shape	(a) 	(a, refined) 
Number of Patches	2	64
Number of Trimming Curves	8	256
% Far-field Evaluation	11.24%	99.31%
% Near-field Evaluation	45.37%	0.67%
% Edge Case Evaluation	43.38%	0.002%
Avg. Time across Query Points	0.0299	0.000523
Avg. Time for Far-field	1.125×10^{-5}	0.000399
Avg. Time for Near-field	0.000155	5.77×10^{-5}
Avg. Time for Edge Cases	0.0297	6.645×10^{-5}

Figure 6.16: All times in seconds. Comparative results between the original surface and its refined equivalent. Notably, reducing the size of each patch dramatically reduces the distances at which edge cases need to be recorded, and so we are able to more effectively utilize our memoization strategy for far-field query points.

In brief, we find intersections between a line and an untrimmed NURBS patch by recursively subdividing the patch and removing portions of the untrimmed surface where intersections cannot occur, verified through a bounding box of the surface. As a base-case for recursion, we treat patches which are "approximately bilinear" (i.e., have control points which are nearly coincident with the

surface along a uniform grid) using the GARP algorithm (Geometric Approach to Ray/bilinear Patch) [140], which describes a closed-form solution for the intersection of a line with a bilinear patch. Once an intersection with the bilinear patch is recorded, its coordinates in parameter space are mapped back to the parameter space of the original surface. When all intersections with the untrimmed patch are recorded, we purge intersection points which occur outside the parameter space defined by the trimming curves, evaluated via the 2D GWN for improved robustness [163]. The full algorithm is presented in Algorithm 9.

The use of this algorithm exposes the parameter ϵ_{ls}, which determines when a sub-patch is considered to be approximately bilinear (See Algorithm 11). This parameter naturally influences both the accuracy and the precision of the line-surface intersection routine, as increasing ϵ_{ls} means the base-case bilinear patch may diverge more from the ground-truth geometry. However, decreasing ϵ_{ls} also increases the number of recursive subdivisions needed to capture the intersection, although we note that this computational cost is somewhat marginal relative to the burden of the recursive quadrature algorithm. In our numerical experiments, we have found a choice of $\epsilon_{ls} = 10^{-6}$ to be reasonable in terms of computational efficiently, while still providing sufficient accuracy and precision for the intersection routine.

With respect to accuracy, it is important that the value of ϵ_{ls} be sufficient that true intersections are not missed, which occurs most frequently when the line is nearly tangent to the surface. While this issue can be mitigated with a more conservative tolerance for ϵ_{ls}, the problem is more easily avoided in the context of the broader GWN algorithm by selecting the line so that it is orthogonal or nearly-orthogonal to the surface at the point of intersection. Directly identifying such a line would require additional processing, such as a closest point query, the cost of which may exceed that of the GWN calculation. It is for this reason that we heuristically select the line in the direction of an approximated average surface normal, which is effective for the majority of non-adversarial CAD surfaces.

On the other hand, the nature of the proposed GWN algorithm is such that, beyond a certain threshold, improving the *precision* of the line-surface intersection routine beyond that which is

sufficient to ensure its *accuracy* does not influence the subsequent containment decision. This is because the intersection point only affects the evaluation of Equation 6.19 through a fixed, half-integer value, and so the line-surface intersection routine only needs to be precise enough to (1) correctly identify which side of the surface the query point is on, and (2) ensure the query point is correctly handled as an edge case.

The first requirement is met by determining surface orientation only *after* the intersection point is mapped back to the parameter space of the original surface. This means that even if the query point is misclassified as being on the wrong side of the bilinear patch, it is only misclassified with respect to the original surface in adversarial cases.

To address the second requirement, we first consider the case where perform a preprocessing step on each untrimmed NURBS surface to linearly extend its parameter space by 1% of the patch's parameter space bounding box [181]. This means the intersection routine can robustly capture "near-misses" with the surface in edge cases where an intersection needs to be recorded, but is not due to numerical error. We also choose a somewhat conservative radius for the disk removed in the treatment of edge cases (also equal to 1% of the patch's parameter space bounding box), which ensures that both the true intersection point and the approximate intersection point are both contained by the same extracted disk. In doing so, the remaining surface will not intersect the line regardless, and the GWN algorithm can proceed as normal.

To demonstrate this insensitivity to the choice of ϵ_{ls}, we evaluate the GWN with respect to the "teardrop" shape in Figure 6.17. The bottom of this shape is spherical, but is composed of 4 degenerate, bicubic patches rather than the biquintic patches used in the example of Figure 6.10. The top shape is generated by rotating a polynomial cubic Bézier curve around the z-axis, which provide inflection points which pose additional challenge for the line-surface intersection test.

Around this shape, 10^5 points are randomly sampled from a uniform distribution in an axis-aligned bounding box. For these points, we consider both the number of misclassifications out of 10^5 across varying levels of ϵ_{ls}, as well as the average wall-clock run-time needed to evaluate the GWN across all points. In this example, we set the tolerance for the adaptive quadrature algorithm

$\epsilon_q = 10^{-6}$. As expected, we see that the number of misclassified points rapidly decreases to zero, while the associated cost of the algorithm increases as more surface subdivisions are needed to evaluate intersection points. We note, however, that this increased computational cost is indeed marginal relative to the total cost of the algorithm.

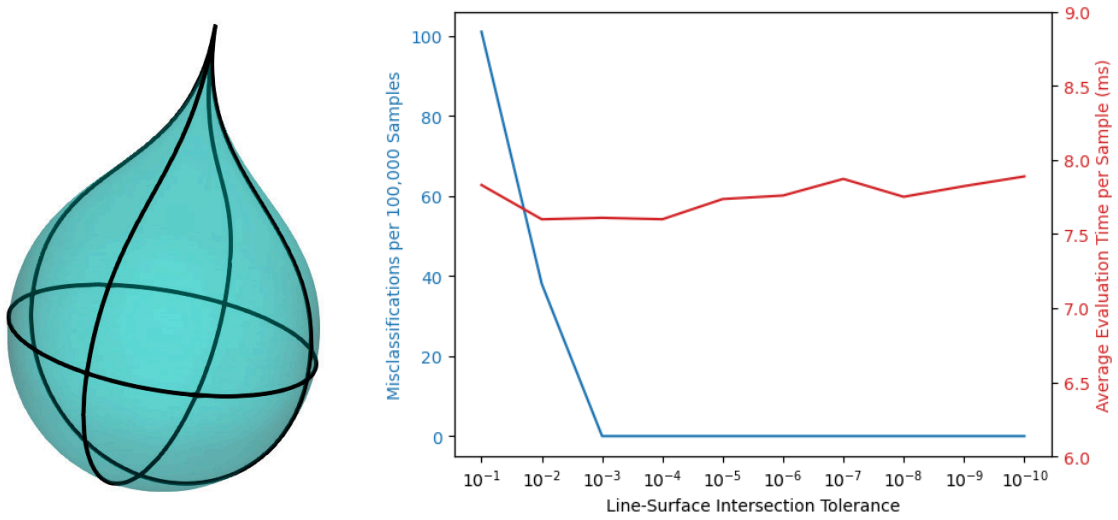


Figure 6.17: We evaluate the 3D GWN on 10^5 query points uniformly sampled from an axis-aligned bounding box across various tolerance levels for ϵ_{ls}. We observe that beyond a certain threshold, there is sufficient accuracy in the line-surface intersection routine to ensure there are no misclassifications among the sampled points.

6.7.0.2 Adaptive Quadrature

To precisely evaluate the line integral in Equation 6.19 requires a numerical integration scheme that is accurate for the near-singular integrand, even if the line integral is numerically stable on a larger domain than the surface integral in Equation 6.5 (See Figure 6.10).

In our implementation, we evaluate the integral to arbitrary accuracy with a recursive subdivision approach, applying a fixed-order Gaussian quadrature rule at each level of recursion. We stop the recursion when the difference between the evaluated integral on the entire curve and the sum of the integrals on the two halves is less than a user-defined tolerance ϵ_q.

Gaussian rules are useful as they are capable of maintaining high accuracy even without

Algorithm 9: `LinePatchIntersection` Find all intersections between a line and a trimmed Bézier or NURBS surface.

Input: L: Line
S: Trimmed Bézier or NURBS surface
ϵ_{ls}: Numerical tolerance for `isApproximatelyBilinear`
Output: t, u, v: Lists of the parameters of intersections

```
   /* Initialize empty lists for intersections                          */
 1 t, u, v ← {}, {}, {}
 2 if not intersects(L, BoundingBox(S)) then
 3 |   return t, u, v
 4 if isApproximatelyBilinear(S, ε_ls) then
   |   /* Find up to two line-bilinear patch intersections             */
 5 |   t, u_0, v_0 ← GARP(S, L, ε)
 6 |   u, v ← MapToOriginalKnotSpace(u_0, v_0)
 7 |   return t, u, v
 8 else if S is a Bézier surface then
 9 |   return t, u, v ← LineBezierIntersectionRecursive(L, S)
10 else
11 |   S_i ← ExtractBezier(S)
12 |   foreach Bézier Patch S_i do
13 |   |   t_0, u_0, v_0 ← LineBezierIntersectionRecursive(L, S_i)
14 |   |   u_0, v_0 ← RescaleToKnotSpace(S, u_0, v_0)
15 |   |   concatenate(t, t_0)
16 |   |   concatenate(u, u_0)
17 |   |   concatenate(v, v_0)
18 |   t, u, v ← RemoveDuplicates(t, u, v, ε)
19 foreach Intersection point u_i, v_i do
20 |   if (u_i, v_i) ∉ TrimmingCurves(S) then
21 |   |   pop(t, t_i) pop(u, u_i) pop(v, v_i)
22 return t, u, v
```

specific structure in the integrand. Our recursive subdivision strategy is specifically chosen because it effectively concentrates quadrature nodes near unstable regions of the integrand, i.e., when the query point is close to the 3D space curve which defines the domain of integration. Elsewhere, where the integrand is stable, we maintain the typically high accuracy of Gaussian quadrature without additional computational expense. To improve the computational performance of our own implementation further, we cache the quadrature nodes and surface tangents at each level of recursion, and reuse these values across multiple queries. The full algorithm is presented in

Algorithm 10: `LineBezierIntersectionRecursive` Recursively find all intersections between a line and an untrimmed Bézier surface. Use approximate bilinearity as a base case for recursion. Uses GARP algorithm for internal line-bilinear patch intersection [140].

Input: L: Line
B: Untrimmed Bézier surface
ϵ_{ls}: Numerical tolerance for `isApproximatelyBilinear`
Output: t, u, v: Lists of the parameters of intersections

if not `intersects`$(L, \texttt{BoundingBox}(S))$ **then**
  return t, u, v
if `isApproximatelyBilinear`(S, ϵ_{ls}) **then**
  `/* Find up to 2 line/bilinear patch intersections via GARP algorithm */`
  $t_0, u_0, v_0 \leftarrow$ `GARP`(S, L)
  $u_0, v_0 \leftarrow$ `MapToOriginalParameterSpace`(u_0, v_0)
  `concatenate`(t, t_0)
  `concatenate`(u, u_0)
  `concatenate`(v, v_0)
  return
$B_1, B_2, B_3, B_4 \leftarrow$ `split`(B)
foreach Bézier Patch B_i **do**
  $t_0, u_0, v_0 \leftarrow$ `LineBezierIntersectionRecursive`(L, B_i, ϵ_{ls})
  $u_0, v_0 \leftarrow$ `RescaleToKnotSpace`(B, u_0, v_0)
  `concatenate`(t, t_0)
  `concatenate`(u, u_0)
  `concatenate`(v, v_0)
return t, u, v

Algorithm 12.

The exposed numerical tolerance ϵ_q in effect determines how many digits of accuracy are achieved in the computed GWN field. To demonstrate how the choice of ϵ_q affects the accuracy and computational efficiency of the algorithm, we repeat the experiment depicted in Figure 6.17 in Figure 6.18, varying ϵ_q during the evaluation of the GWN with respect to 10^5 uniformly sampled points from a bounding box of the "teardrop" shape. In these examples, we set $\epsilon_{ls} = 10^{-6}$. The results of this example most significantly differ from Figure 6.17 in that there is a more clear relationship between algorithm precision and computational performance.

However, we do observe a similar insensitivity with respect to accuracy, as the number of misclassified points is indeed equal to zero for all values less than $\epsilon_q < 0.01$. This is not unexpected,

Algorithm 11: `isApproximatelyBilinear` Returns true if a Bézier patch is approximately bilinear according to the position of its control points

Input: S: Order (p, q) Tensor product Bézier patch
Input: S: Order (p, q) Tensor product Bézier patch
ϵ: Numerical tolerance for distance

$B(u, v) \leftarrow$ Bilinear surface defined by 4 vertex control points

if $p \leq 1$ **or** $q \leq 1$ **then**
  return True
for $i = 0 \dots p$ **do**
  for $j = 0 \dots q$ **do**
    `/* Select the control point with these indices */`
    $B_{i,j} = \texttt{ControlPoint}(B, i, j)$
    `/* Evaluate the bilinear surface on a uniform grid */`
    $\widetilde{B}_{i,j} = B(\frac{i}{p}, \frac{j}{q})$
    if `squared_distance`$(B_{i,j}, \widetilde{B}_{i,j}) > \epsilon$ **then**
      return False
return True

as in principle only a single digit of accuracy is needed to determine containment, as the floating point value of the GWN is mapped to containment in these examples only through rounding to the nearest integer However, we also see that for $\epsilon_q = 0.1$, there is a single recorded misclassification. This suggests that to ensure a correct classification, the numerical tolerance must be chosen low enough to avoid an incorrect agreement among subdivisions of the domain, and for this reason all presented numerical tests use a fixed value of $\epsilon_q = 10^{-6}$.

6.7.0.3 Disk Extraction

Handling edge cases efficiently requires a specific type of surface subdivision in which the surface is split according to a disk defined in parameter space around a given point (See Figure 6.19). We accomplish this through through the insertion of extra trimming curves around the given point. To the original surface, we add an extra trimming loop around each point in parameter space, and remove all existing trimming curves within the loop. To a copy of the original surface, we remove

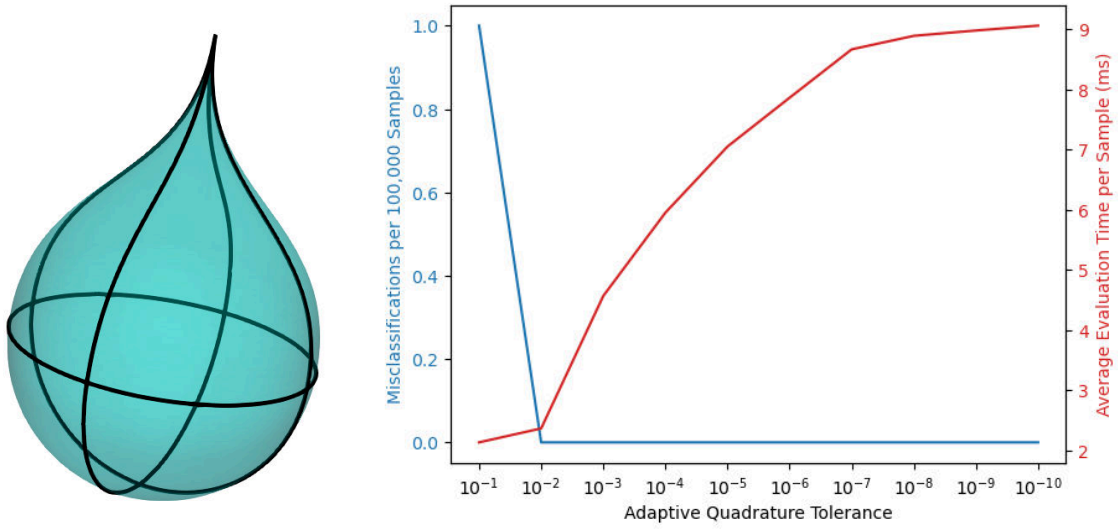


Figure 6.18: We evaluate the 3D GWN on 10^5 query points uniformly sampled from an axis-aligned bounding box across various tolerance levels for ϵ_{ls}. We observe that beyond a certain threshold, there is sufficient accuracy in the line-surface intersection routine to ensure there are no misclassifications among the sampled points.

Algorithm 12: `EvaluateLineIntegral` Numerically evaluate the integral in Equation 6.19 with an adaptive Gaussian quadrature rule of fixed order. For a fixed curve and recursive depth, the quadrature nodes are cached and reused across queries.

Input: C: Trimming curve
ϵ_q: Numerical tolerance for stopping criterion

```
/* Assumes that the query point is the origin                    */
w = GaussianQuadrature(C)

if C is sufficiently far from the origin then
    return w
C_1, C_2 ← bisect(C)
w_1 = GaussianQuadrature(C_1)
w_2 = GaussianQuadrature(C_2)
if |w − (w_1 + w_2)| < ε_q then
    return w_1 + w_2
else
    /* Repeat the quadrature on each half                        */
    return EvaluateLineIntegral(C_1, ε_q) + EvaluateLineIntegral(C_2, ε_q)
```

all trimming curves *exterior* to the loop in parameter space. This processing is greatly simplified by the use of 2D generalized winding numbers in the calculation of visibility within the parameter

space of the surface, as these trimming curves need not be explicitly connected to form closed loops.

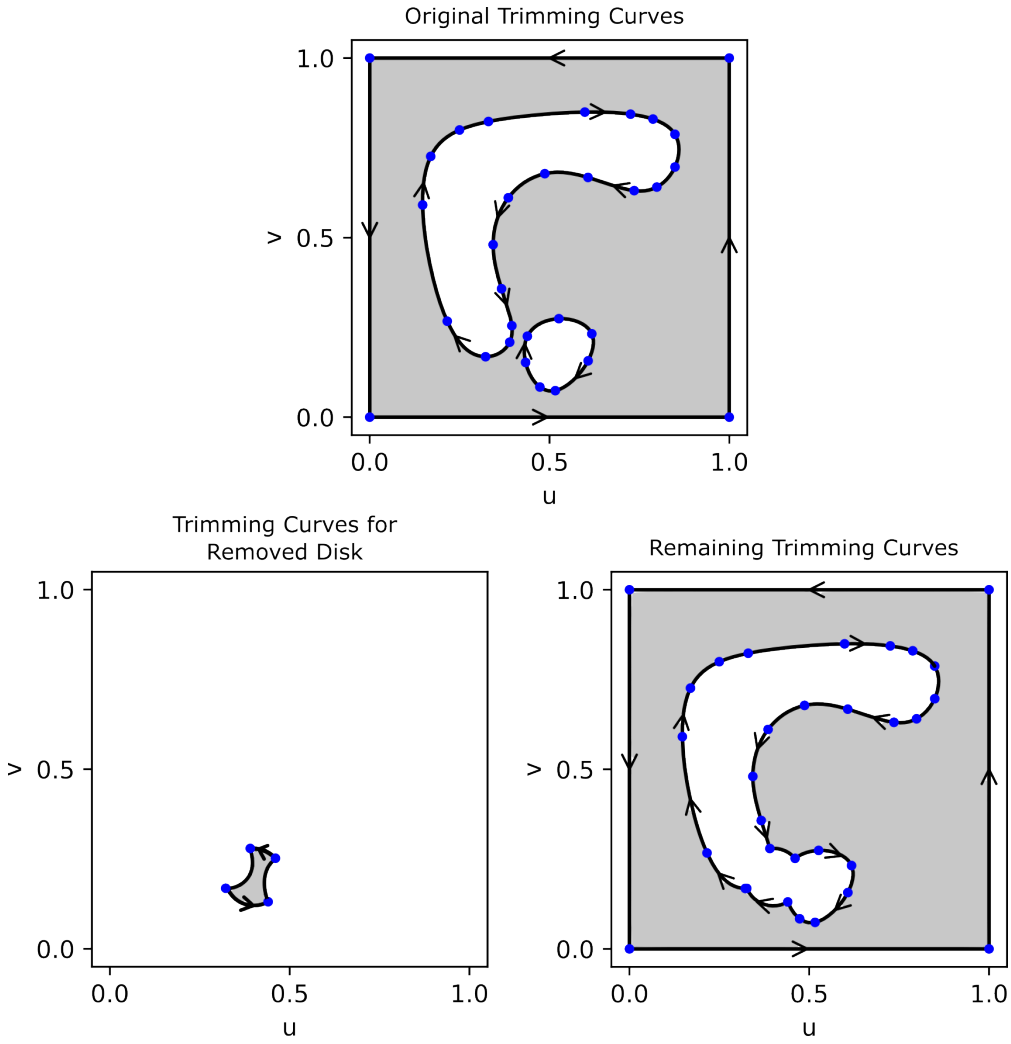


Figure 6.19: We show the effect of disk extraction as applied to the edge case in Figure 6.9(a).

6.7.0.4 Evaluation of 2D GWN via Stokes' Theorem

In the previous chapter, we have presented a method of evaluating the 2D GWN field through a purely geometric routine based on successive curve subdivision as a method of altogether avoiding the use of numerical quadrature rules. A natural question then, is if the proposed evaluation method of the 3D GWN via Stokes' theorem can be similarly applied the 2D problem domain as an alternative method of avoiding unstable numerical integration.

To consider this, we revisit the integral formulation of the 2D GWN in Equation 6.2, which defines the 2D GWN for a query point at the origin as

$$w_C = \frac{1}{2\pi} \int_C \frac{x \cdot \hat{n}}{||x||^2} \, dx. \tag{6.19}$$

Expressed as an integral over a vector field, we have

$$\begin{aligned} w_C &= \frac{1}{2\pi} \int_C \left\langle \frac{-y}{x^2+y^2}, \frac{x}{x^2+y^2} \right\rangle \cdot \, d\vec{r}, \\ &= \frac{1}{2\pi} \int_C \langle P, Q \rangle \, d\vec{r}. \end{aligned} \tag{6.20}$$

To apply the fundamental theorem of line integrals, the specification of Stokes' theorem to a 2D curve, requires defining a potential function F, for which $\nabla F = \langle P, Q \rangle$, which is most directly satisfied by the function

$$F(x, y) := \arctan\left(\tfrac{y}{x}\right), \tag{6.21}$$

which would suggest the 2D GWN to be evaluated simply as

$$\begin{aligned} w_C &= \frac{1}{2\pi} \left(F(C(1)) - F(C(0)) \right) \\ &= \frac{1}{2\pi} \left(\arctan\left(\tfrac{C(1).y}{C(1).x}\right) - \arctan\left(\tfrac{C(0).y}{C(0).x}\right) \right) \end{aligned} \tag{6.22}$$

where $C(0), C(1) \in \mathbb{R}^2$ are the endpoints of the curve. In many ways, this particular formulation is the most direct analogue to the evaluation of the GWN via Equation 6.11. However, as is the case in 3D, such a formulation clearly cannot define the GWN at arbitrary points in space, as the endpoints provide no specific information about the internal geometry of the curve. To see how this formulation impacts evaluation of the GWN field, we follow the example of Figure 6.7, and compare the results of Equation 6.22 to the ground truth GWN.

As in 3D, the fundamental mathematical issue with this formulation is that the arctan used for the potential function has a branch cut which interferes with the calculation for certain query points. Importantly, we see in Figure 6.20 that this is not universally a concern, as query points which are "far-away" from the curve (in the sense that the query is to the left or to the right of

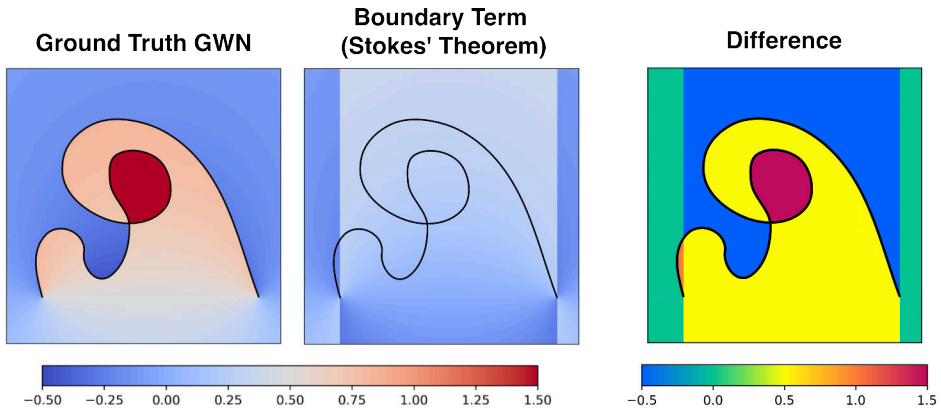


Figure 6.20: In 2D, the difference between the GWN for a curve (left) and the value of a particular Stokes' theorem derived boundary term (middle) is always half-integer valued, and partitioned by the geometry (right). This is analogous to the 3D case in Figure 6.8, although this particular formulation is not as useful for evaluating the 2D GWN as the framework presented in Chapter 5.

both curve endpoints) do indeed produce the correct value. This is exactly the observation made for our 3D application of Stokes' theorem, and in both cases the correct value can be obtained through a correction term which accounts for the number of intersections of a particular line with the surface.

Furthermore, the 2D vector field also permits use of a different potential function, each of which introduces a different branch cut that would, in principle, influence the correction term added For example, if we substitute the arctan in Equation 6.21 for the `atan2` function, then necessary correction term accounts for line-curve intersections not in the direction of the $y-$axis, but rather the $x-$axis.

In any case, the formulation provided by Equation 6.22 is closely associated with the angle subtended by the line defined by the curve endpoints at the query, i.e., the GWN of the closure. From this perspective, the technique suggested in the previous chapter is reframed as a way of correcting the result of *some* Stokes' theorem formulation through an integer containment query test, rather than a line-curve intersection test. In some sense, there is a computational advantage to the method of the previous chapter as well, as the fractional component of the GWN can be evaluated with a single arccosine, rather than two calls to arctangent. It is for this reason that we

consider the more strictly geometric solution presented in the previous chapter to be the superior solution strategy, and consider the problem of finding a more strictly geometric solution to the 3D problem an interesting avenue for future work.

6.8 Conclusions

In this work we have demonstrated our algorithm for efficiently and reliably computing the generalized winding number field defined by unstructured collections of trimmed NURBS patches. This permits the definition of a robust containment query that can be applied to general CAD models, even in the presence of non-watertightness and other geometric errors.

For points which are far from the surface, we evaluate the relevant surface integral using a novel application of Stokes' Theorem, reducing the computational complexity needed to evaluate the GWN field to a collection of line integrals defined by trimming curves. For points which are near to the surface, the same reformulation into line integrals is applied, but importantly is paired with a procedure to adjust the numerical value so that it exactly maintains the geometric fidelity of the input model.

Although the focus of this work is primarily the theoretical underpinnings of our Stokes' theorem reformulation and its implications for the evaluation of the GWN for individual surface patches, we are interested in applying hierarchical approximation methods as in [10] to improve batch performance across collections of patches, particularly in the case of far-field evaluation. Similarly, we are interested in potential performance improvements that come about from the use of more specialized integration techniques, such as those described in Klintegerg et al. [90], which have been demonstrated as suitable for evaluating the 2D GWN field more efficiently than generic quadrature methods.

Finally, we are very interested in further exploring the problem of robustly handling trimming curves. For our algorithm to function properly, we require that the trimming curves for each patch are "well-posed," in the sense that they properly define an enclosed region in the parameter space of the patch. While this requirement is somewhat relaxed by the use of the 2D GWN to evaluate

containment in the trimming curves, it is still occasionally not met by trimming curves generated by CAD software (See Figure 6.21).

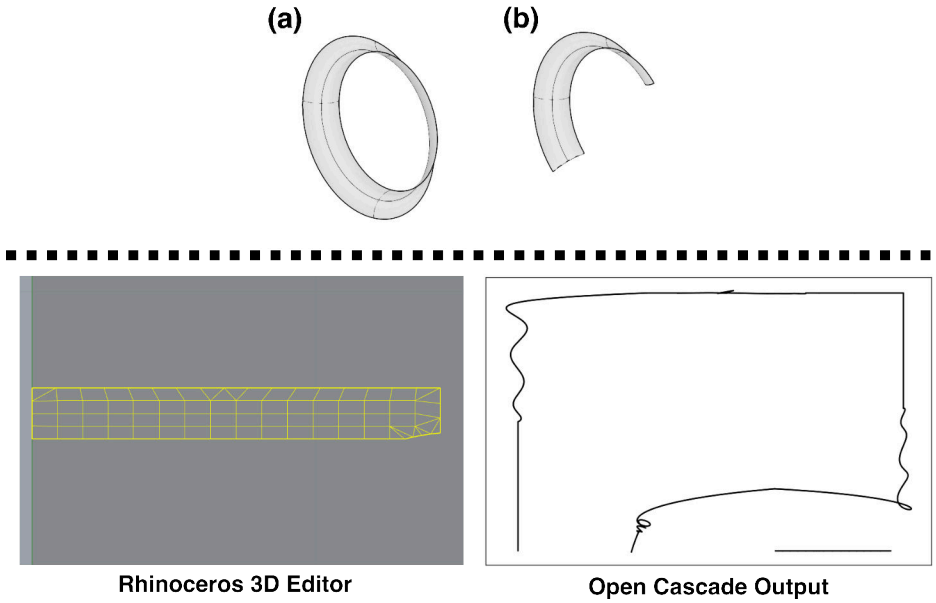


Figure 6.21: We show an untrimmed NURBS patch (a) along with its trimmed counterpart (b). A common problem in computer graphics is a lack of inoperability between different CAD systems. In this case, the Rhinocerous 3D modeling engine considers the trimming curves in a way that is sensible for the patch, but exporting the trimming curves directly from the .STEP file produces messy geometry that cannot be immediately resolved through a robust containment query.

Chapter 7

3D Multi-plane Moment-of-Fluid Interface Reconstruction

7.1 Abstract

Moment-of-fluid (MOF) methods for interface reconstruction approximate the region occupied by material in each mesh element only through reference to its geometric moments. We present a 3D MOF method that represents the material (POM) in each cell as the convex intersection of the cell and multiple half-spaces, each selected to minimize the least-squares error between computed moments of the approximated material and provided reference moments.

This optimization problem is highly non-linear and non-convex, making the numerical result very sensitive to the initial guess. To create an effective initial guess in each cell, we construct an ellipsoid from 0^{th}-2^{nd} order reference moments such that its shape corresponds with that of the POM. Within this ellipsoid we inscribe a polyhedron, and initialize the minimization problem with the half-spaces defined by each of its faces. This polyhedron has minimally 4 faces, and using up to 3^{rd} order moments permits optimization over up to 20 unknown values. We therefore define MOF methods that utilize 4, 5, or 6 half-spaces, correspondingly initialized with the faces of a single inscribed tetrahedron, triangular prism, or hexahedron. Stability of the non-linear optimization is further improved with a prepossessing step that normalizes the reference moments according to the axes of the reference ellipsoid. Using this approach, the non-linear least-squares solver reliably converges to a near-global minimum from a single initial guess.

In collaboration with Mikhail Shashkov

7.2 Introduction

7.2.1 Background and Rationale

In simulating the flow of multiple materials, their shape must be accurately captured over a discrete computational mesh. The goal of material interface reconstruction is then to use data provided by the simulation of this flow to approximate the intersection of the region occupied by material with each mesh element—the per-cell pieces of material (POM)—while conserving the volume of each POM.

Among volume tracking techniques, methods for piecewise linear interface calculation (PLIC) are widely used for their simplicity and utility in adjacent applications. Such methods define the interface in each cell with a single linear element whose position is informed by the tracked volume, thereby enforcing a physically conservative solution. At the same time, the orientation of this segment is chosen to resemble that of the underlying material fragment. In the volume-of-fluid family of methods, the orientation is informed by the volume fraction of neighboring cells, such as via an estimate of the gradient of the volume fraction field [134, 106], a coupled level-set method [171], or more recently methods of machine learning [22].

There is also longstanding interest in VOF methods which represent the per-cell interface with more complex geometry, which serves to increase their accuracy or permit more direct analysis of derived quantities like curvature. For example, there are 2D PLIC methods which represent the interface as a single "bent" line [145, 101], or even multiple disjoint segments [102, 66]. Other methods utilize high-order curves and surfaces to approximate the interface, such as parabolas and paraboloids [135, 45, 44], splines [53], or circular arcs and spheres [117, 104, 27].

However, in all of these examples, the placement of the interface is necessarily informed by the data of neighboring cells. In contrast, a key feature of moment-of-fluid (MOF) methods is that each POM is approximated using only data from the cell it occupies. In particular, we consider the geometric moments of the piece of material, the most general geometric characteristics of any shape [51].

In this work, we describe a new MOF method for 3D interface reconstruction, for which we assume that the moments of each POM up to some order (referred to hereafter as reference moments) are given to us according to the specific setting. As a motivating example, we consider hydrodynamic simulation. At the initial time, moments can be obtained from the known initial material shape by its exact or approximate intersection with cells of the mesh. At later times, the material moments have been advanced in time according to the governing physics of the problem, the simplest process being advection [158, 2]. In this work, however, we consider the origin of these moments to be irrelevant to their use in the described MOF methods.

In general, the MOF approach for interface reconstruction defines an approximation of the POM by intersecting the computational cell with some shape. These shapes are parameterized in such a way that the difference between the moments of their intersection with the cell and that cell's reference moments can be minimized via numerical optimization. As the prototypical example, the original MOF-PLIC method in 2D uses the material volume (zeroth-order moment) and centroid (ratios of first-order and zeroth-order moments) to determine parameters of a single half-plane [39]. These parameters are selected to minimize the error between the reference centroid and the computed centroid of the material polygon approximates the POM in the cell.

This methodology extends readily to 3D, using the intersection of a single half-space and the cell to approximate the material polyhedron for each POM [2, 114]. It also allows for extensions that approximate the POM with a more complex interface, again using only local data. Such extensions in 2D include methods that utilize circular arcs [157], more general quadratic curves [164, 139], or multiple linear segments [158, 68, 31]. In cases where the interface has more parameters than a single plane, higher-order moments can be incorporated in the error function to ensure that the ensuing minimization problem is not underdetermined.

7.2.2 Motivation

There are many incentives to develop computationally efficient interface reconstruction methods in 3D, particularly those that utilize local information to reproduce geometrically complex

material. For example, the exclusive use of per-cell data makes such a method more robust to unstructured computational meshes. Furthermore, the computational benefits of a parallelizable method like MOF become even more pronounced as the number of cells increases with spatial dimension.

However, there is considerably more variety in the kinds of geometric features that cannot be adequately captured by the single plane used in a typical MOF-PLIC method, several of which we categorize in Figure 7.1. Many of these features are described by extruding their 2D analogues along the third dimension, such as "wall"-type shapes and edge-corners that require two planes to be reproduced exactly. Others are more unique to 3D space, such as 3D filaments and tips, as well as vertex-corners, each of which require three planes to accurately reproduce. Finally, four planes are necessary to adequately approximate regions that are entirely embedded within a computational cell. With these shapes in mind, we target a MOF method that represents the interface by the convex intersection of, at minimum, four half-spaces. To our knowledge, there exist no contemporary methods of performing a 3D MOF interface reconstruction with interface geometry more complex than a single plane.

We prioritize a reconstruction method with linear boundary components because doing so ensures it is compatible with adjacent physics applications. For example, performing dynamic reconstructions through a Lagrangian pre-image requires computing intersections between material and arbitrary backtraced computational cells, and so there is little fundamental difference applying such a technique to a single plane (as in [2]) and in the proposed multi-plane technique. Similarly, such a method generalizes far more readily to multi-material scenarios using the strategy of nested dissection described in [2], as exact moments can be computed from the intersection of arbitrarily many material interfaces. This is often not the case for methods which consider curved interfaces: even if one can compute exact moments for the intersection of the computational cell and a sphere [170] or paraboloid [44], the problem becomes intractable when even two separate curved interfaces are considered. We consider the specific implementation of the proposed MOF method within these applications to be future work.

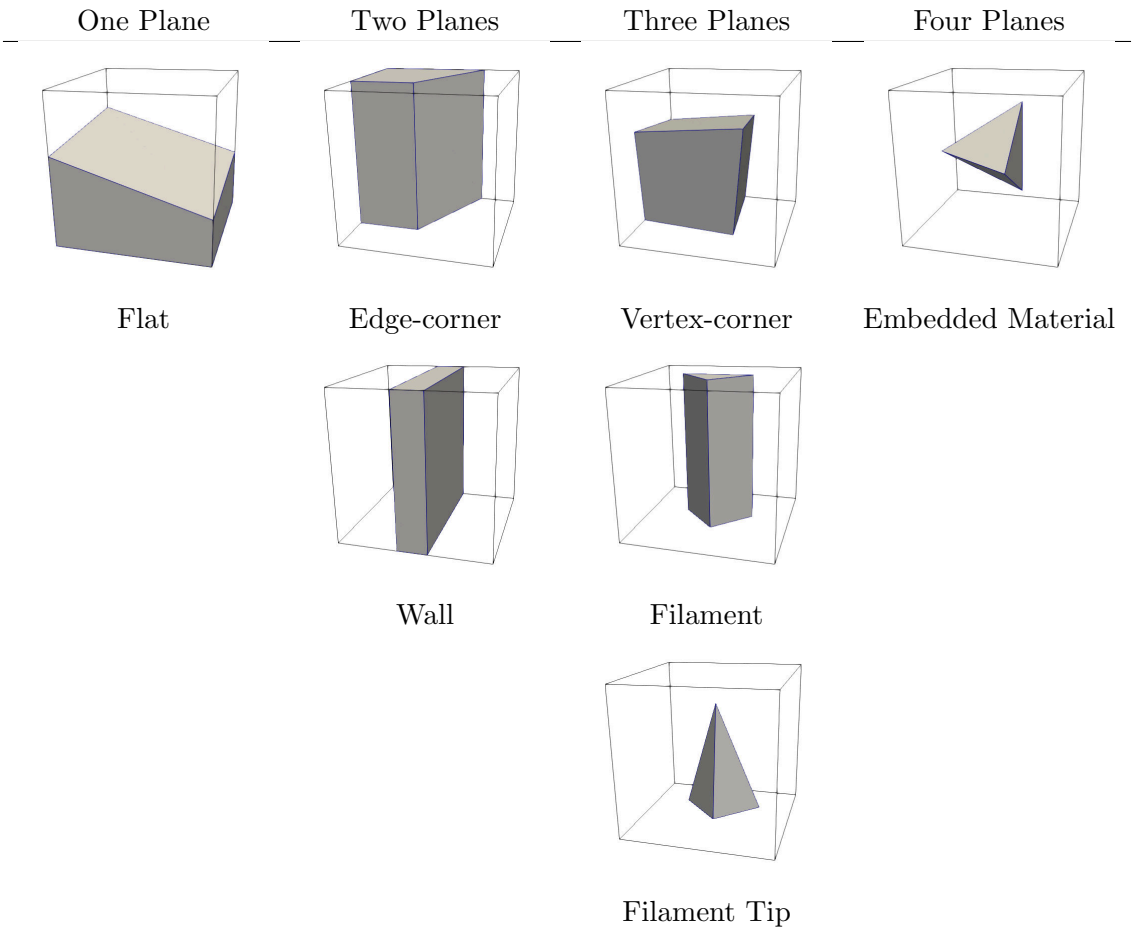


Figure 7.1: Shape types that can be described by up to four planes. We target a method that can reproduce these types of geometric features

As in the typical MOF approach, we formulate interface reconstruction as a non-linear optimization problem that finds a representation of the approximate POM to best match its computed moments to some provided reference moment data.

As an aside, we follow the notation of [157] and distinguish between MOF methods with subscripts and superscripts. For example, consider the MOF^2_{2hp} method for 2D interface reconstruction, where the superscript represents the maximum order of the moments considered, and the subscript represents the type of bounding geometry used, in this case 2 half-planes. Despite the slight abuse of notation, it will be clear from context whether the method is designed for use 2D or 3D space. For example, MOF^2_{2hp} is assumed to be a 2D method using two half-planes. The

analogous 3D method is MOF^2_{2hs}, which uses the same order of moments, but instead bounds the POM by two 3D half-spaces.

In MOF^2_{2hp} and many derived 2D techniques, an ellipse is used to create the initial guess for the relevant non-linear optimization problem [158]. A unique reference ellipse can be constructed from moment data up to second-order so that its volume and centroid coincide with the material, and so that the length and orientation of its main axes correspond with the principle directions and magnitude of elongations of the ground-truth material. Although the construction of this reference ellipse generalizes readily to three spatial dimensions in the form of a reference ellipsoid, the analogous *usage* is necessarily different in 3D, in large part due to same complexities of 3D space that motivate high-order MOF methods in the first place.

The least-squares error between computed moments and reference moments used in the objective function of many MOF methods is often both highly non-linear and non-convex as a function of the chosen parameterization, and so typical descent methods tend towards local minima if the initial guess is far from the desired global minimum. This makes the numerical result of the optimization highly sensitive to the choice of initial guess. To compensate for this, the final interface in MOF^2_{2hp} is generated through repeated solutions of the optimization problem, each of which is initialized using a pair of edges from a polygon inscribed in the reference ellipse. the intersection of one pair with the cell will sufficiently approximate the material so that the subsequent descent algorithm convergences to the true global minimum.

While this is already an expensive process in 2D, the analogous procedure in 3D is largely infeasible. This is due to the additional combinatorial burden of selecting faces from an inscribed polyhedron combined with the increased cost of solving the non-linear optimization. A possible alternative to the combinatorial approach in 2D is presented in the MOF^2_{PIE} method of [31], where the interface is instead represented by the intersection of the computational cell with *all* half-planes defined by the edges of the polygon inscribed in an ellipse. The position of the polygon is fixed relative to the ellipse, and then the ellipse's 5 parameters are optimized in a typical MOF fashion, minimizing the difference between computed and reference moments. This permits the use

of a single initial guess for the non-linear optimization, the reference ellipse itself. This approach potentially generalizes directly to 3D, with an interface represented by a polyhedron inscribed in a 9 parameter ellipsoid, initialized using reference moments. However, such a method is limited by the large number of configurations possible for the connectivity of vertices and edges for general polyhedron, making it difficult for the same initial guess to converge to all possible geometric features of interest.

It is through comparison to these methods that we motivate our proposed strategy for interface reconstruction, guided by a desire to represent complex 3D geometry with as few solutions of the non-linear optimization problem as possible. As in MOF^2_{PIE}, we initialize the optimization procedure with an interface represented by the faces of a polyhedron inscribed in the reference ellipsoid, each of which define a half-space in the intersection with the cell. However, rather than parameterizing the optimization through the parameters of the circumscribing ellipsoid, we instead optimize over the full set of parameters of each plane defined by the polyhedron.

Properly defining a closed, inscribed polyhedron requires at minimum 4 faces to form a tetrahedron, which is also conveniently sufficient to appropriately recreate our target geometries depicted in Figure 7.1. However, this results in a non-linear least-squares problem with minimally 12 parameters, which is underdetermined when only the 10 zeroth- to second-order moments are considered. To compensate for this, we instead minimize the least-squares error over the 20 moments up to third order. This fully describes the MOF^3_{4hs} method, which utilizes 3^{rd} order moments to construct an interface out of 4 half-spaces, initialized from a single tetrahedron inscribed in the reference ellipsoid. At the same time, using third-order moments also permits extension to a 15 parameter MOF^3_{5hs} method and a 18 parameter MOF^3_{6hs} before the optimization problem becomes underdetermined. Incorporating higher-order moments is also useful from a purely geometric perspective, as third-order moments encode a greater amount of information about the material's orientation, a fact that has motivated [31] to consider a 2D MOF^3_{PIE} method alongside MOF^2_{PIE}.

To clarify the objectives of this work, we expect the proposed family of methods to reliably reproduce shapes of the types depicted in Figure 7.1, most of which can be theoretically described

by only three planes. However, in cases where the shape is described by four or more planes, we claim that it is only *likely* for the proposed family of methods to achieve an exact reconstruction from a single initial guess. Ultimately, this technique utilizes an important trade-off between computational cost and reliability interface reconstruction. Each of MOF^3_{4hs}, MOF^3_{5hs}, and MOF^3_{6hs} solve an increasingly expensive minimization problem, one which uses more half-spaces than would be necessary to reproduce the target shapes in Figure 7.1. Yet the use of additional, intentionally redundant half-spaces in the optimization avoids the even greater cost of repeating the minimization procedure with different initial guesses to more confidently reach the global minimum.

This paper proceeds as follows. In Section 7.3, we present an overview of the relevant details on 3D moments including formulas related to the translation, scaling, and rotation transformations. These are used to normalize given moment data and in the construction of the reference ellipsoid. Next, in Section 7.4 we discuss details of the new MOF interface reconstruction method. This includes definition of the objective function for non-linear optimization, our choice of initial guess, and the customized Levenberg-Marquardt algorithm used, among other details. We summarize the proposed MOF algorithms in Section 7.5. We present numerical results in Section 7.6, including the results of reconstruction on a single computational cell as well as across the full mesh. Discussion related to practical implementation is presented in Section 7.7. This includes the reconstruction of non-convex material, the appearance of geometric artifacts, 3D polyhedra with identical moments up to some order, and sensitivity to noise in reference moment data. Conclusions and a discussion on future work to be done is in Section 7.8. Finally, acknowledgements are given in Section 7.9.

7.3 Moments Primer

7.3.1 Translation, Scaling, and Rotation of Volume Moments

To introduce the necessary notation, we restate the objective of the proposed method as finding an approximation of the material Ω within a convex computational cell C. We denote this approximation as Ω_r.

In this work, we consider standard 3D geometric moments of Ω given by

$$M_{ijk} = \int_{\Omega} x^i y^j z^k \, dV. \tag{7.1}$$

These "raw" moments describe important geometric features of the material, such as the volume M_{000} and centroid $\vec{c} = (M_{100}/M_{000}, M_{010}/M_{000}, M_{001}/M_{000})$. However, it is useful and often necessary to consider alternate formulations and transformations of moment data. For example, consider a shift transform $S(s_x, s_y, s_z) : \Omega \to \Omega'$. If the material is shifted by $\vec{s} = (s_x, s_y, s_z)$ according to $\widehat{\Omega} = \Omega - \vec{s}$, then its moments can be computed as

$$M'_{ijk} = \int_{\Omega - \vec{s}} x^i y^j z^k \, dV = \int_{\Omega} (x - s_x)^i (y - s_y)^j (z - s_z)^k \, dV. \tag{7.2}$$

Importantly, this latter formulation allows one to compute these moments directly from their raw counterparts without reference to the underlying material Ω. For example, moments up to second order are given by

$$\begin{aligned}
M'_{000} &= M_{000} \\
M'_{100} &= M_{100} - s_x M_{000} \\
M'_{010} &= M_{010} - s_y M_{000} \\
M'_{001} &= M_{001} - s_z M_{000} \\
M'_{200} &= M_{200} - 2 s_x M_{100} + s_x^2 M_{000} \\
M'_{020} &= M_{020} - 2 s_y M_{010} + s_y^2 M_{000} \\
M'_{002} &= M_{002} - 2 s_z M_{001} + s_z^2 M_{000} \\
M'_{110} &= M_{110} - s_x M_{010} - s_y M_{100} + s_x s_y M_{000} \\
M'_{101} &= M_{101} - s_x M_{001} - s_z M_{100} + s_x s_z M_{000} \\
M'_{011} &= M_{011} - s_z M_{010} - s_y M_{001} + s_y s_z M_{000}.
\end{aligned} \tag{7.3}$$

Similarly, one can apply a scaling $L(l_x, l_y, l_z) : \Omega \to \Omega'$ to moment data along each coordinate axis by (l_x, l_y, l_z) using the transform

$$M'_{ijk} = l_x^{1+i} \, l_y^{1+j} \, l_z^{1+k} \cdot M_{ijk} \tag{7.4}$$

Finally, we also consider *rotations* around the origin, $R(\alpha, \beta, \gamma) : \Omega \to \Omega'$ defined by the following rotation matrix:

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} c_\beta c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ c_\beta s_\gamma & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma \\ -s_\beta & s_\alpha c_\beta & c_\alpha c_\beta \end{bmatrix}, \tag{7.5}$$

where α, β, and γ are rotations around the x-, y-, and z-axes respectively. Written explicitly, this gives the transformation

$$\begin{aligned} M'_{ijk} &= \int_{R(\Omega;\alpha,\beta,\gamma)} x^i y^j z^k \, dV \\ &= \int_\Omega (c_\beta c_\gamma x + (s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma) y + (c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma) z)^i \cdot \\ &\qquad (c_\beta s_\gamma x + (s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma) y + (c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma) z)^j \cdot \\ &\qquad (-s_\beta x + s_\alpha c_\beta y + c_\alpha c_\beta z)^k \, dV \end{aligned} \tag{7.6}$$

Under this transformation, the rotated first-order moment data are given by

$$\begin{aligned} M'_{000} &= M_{000} \\ M'_{100} &= c_\beta c_\gamma M_{100} + (s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma) M_{010} + (c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma) M_{001} \\ M'_{010} &= c_\beta s_\gamma M_{100} + (s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma) M_{010} + (c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma) M_{001} \\ M'_{001} &= -s_\beta M_{100} + s_\alpha c_\beta M_{010} + c_\alpha c_\beta M_{001}. \end{aligned} \tag{7.7}$$

Higher order moments can be computed similarly. We also note for future use that the inverse transformation $R^{-1}(\alpha, \beta, \gamma)$ has the same form as Equation 7.6, but with the inverse of the rotation matrix in Equation 7.5.

7.3.2 Reference Ellipsoid and Normalization

To ensure stability during numerical optimization, we normalize the reference moments M^r_{ijk} with the transformations described in Section 7.3.1 such that the underlying material Ω is translation-invariant and scale-invariant. This is a critical step for the proposed MOF methods,

as it makes subsequent optimization steps far more robust to unimportant features of the material, such as spatial position and size within the cell.

As our first transformation, we create "central moments" $\overline{M}^r_{ijk}$ which correspond to the reference moments M^r_{ijk} shifted by the centroid $\vec{c} = (M^r_{100}/M^r_{000}, M^r_{010}/M^r_{000}, M^r_{001}/M^r_{000})$ according to Equation 7.2. Under this transformation, we can then consider other useful properties of the second-order central moments, which encode information about the orientation of the material. Specifically, we define a covariance matrix for the material in terms of its central moments, given by

$$A = \frac{1}{\overline{M}^r_{000}} \begin{bmatrix} \overline{M}^r_{200} & \overline{M}^r_{110} & \overline{M}^r_{101} \\ \overline{M}^r_{110} & \overline{M}^r_{020} & \overline{M}^r_{011} \\ \overline{M}^r_{101} & \overline{M}^r_{011} & \overline{M}^r_{002} \end{bmatrix}. \tag{7.8}$$

This matrix is necessarily symmetric and positive semi-definite, and so it defines a quadratic form $x^T Ax$ whose zero level set $x^T Ax = 0$ corresponds to a *reference* ellipsoid in $\mathbb{R}^3$. Historically, quadratic forms of this kind have been used in 2D to define effective initial guesses for the non-linear optimization problem, as the position and orientation of the derived ellipse roughly corresponds to the position and orientation of the underlying material [158, 31]. In this work, we normalize the material according to the corresponding reference ellipsoid through both a rotation transformation and a scaling transformation. Done properly, the centered material $\overline{\Omega}$ and its corresponding moments $\overline{M}^r_{ijk}$ are normalized so that the transformed moments $\widehat{M}^r_{ijk}$ of the transformed $\widehat{\Omega}$ generate a diagonal covariance matrix, and therefore a spherical reference ellipsoid.

To this end, let $\lambda_0, \lambda_1, \lambda_2$ be the three eigenvalues of A with associated eigenvectors $v^{(0)}, v^{(1)}, v^{(2)}$. These eigenvectors can be used to define rotation angles

$$\alpha = \text{atan2}(v^{(1)}_z, v^{(2)}_z),$$

$$\beta = \text{asin}(-v^{(0)}_z),$$

$$\gamma = \text{atan2}(v^{(0)}_y, v^{(0)}_x).$$

These angles are selected so that, when used in the *inverse* rotation transformation $R^{-1}(\alpha, \beta, \gamma)$,

the orientation of the centered material $\overline{\Omega}$ is "undone", and the resulting intermediate material is aligned with the Cartesian axes. Furthermore, each axis-length a, b, c of this ellipsoid is proportional to the square root of an eigenvalue of A, which permits a scaling of each such that the respective lengths are equal. At the same time, we resolve the ambiguity in the constant of proportionality by selecting l_x, l_y, and l_z so that the result is also normalized to have unit material volume $\widehat{M}^r_{000}$. Together, this results in the following scaling coefficients l_x, l_y, and l_z:

$$l_x = \frac{1}{\sqrt{\lambda_0}} \left(\frac{\sqrt{\lambda_0 \lambda_1 \lambda_2}}{\overline{M}^r_{000}} \right)^{1/3}, \quad l_y = \frac{1}{\sqrt{\lambda_1}} \left(\frac{\sqrt{\lambda_0 \lambda_1 \lambda_2}}{\overline{M}^r_{000}} \right)^{1/3}, \quad l_z = \frac{1}{\sqrt{\lambda_2}} \left(\frac{\sqrt{\lambda_0 \lambda_1 \lambda_2}}{\overline{M}^r_{000}} \right)^{1/3}. \tag{7.9}$$

By construction, applying these to the reference central volume moment $\overline{M}^r_{000}$ through Equation 7.4 results in

$$\widehat{M}^r_{000} = l_x l_y l_z \overline{M}^r_{000} = \frac{1}{\sqrt{\lambda_0 \lambda_1 \lambda_2}} \left(\frac{\sqrt{\lambda_0 \lambda_1 \lambda_2}}{\overline{M}^r_{000}} \right) \overline{M}^r_{000} = 1 \tag{7.10}$$

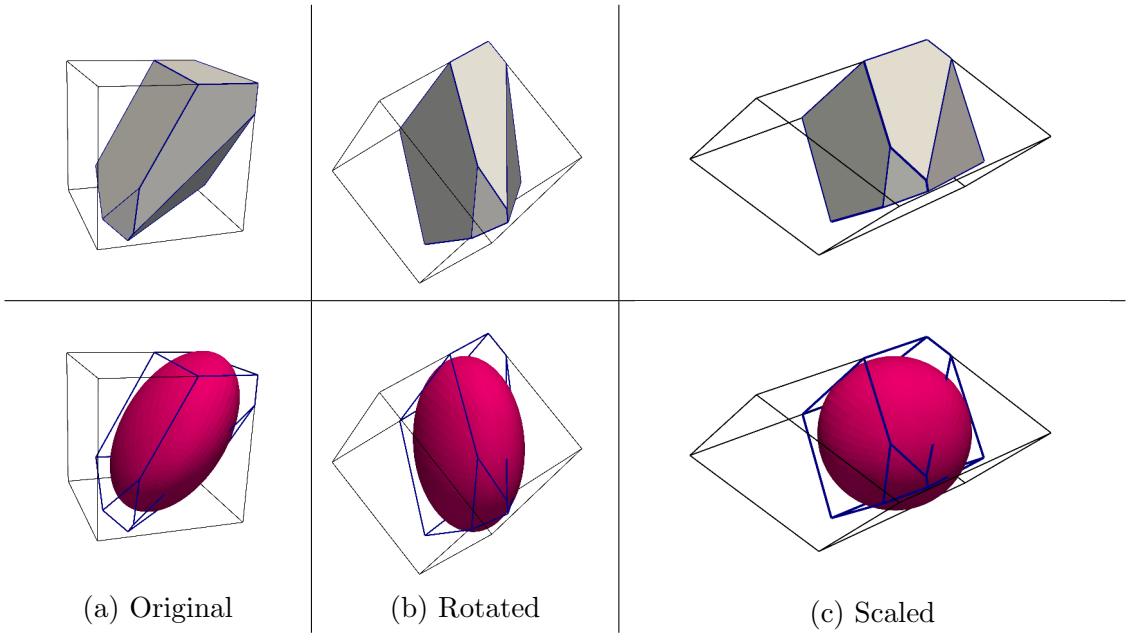
as desired.

Taken together, this sequence of transformations turn the material Ω into $\widehat{\Omega}$ that is centered at the origin, has unit volume, and is not stretched along any particular axis. We can see an example of each stage of this normalization in Figure 7.2. From another perspective, we can consider this mapping to apply not to Ω, but to the entire space in which it is embedded. This means that during optimization, the *raw* moments of the reconstruction are matched to the *raw* moments of $\widehat{\Omega}$, since these are equal to the centered, rotated, and scaled moments of Ω by construction. Importantly, this makes the process of "normalization" wholly disjoint from any manipulation of the objective function (see Section 7.4.1). After the optimal reconstruction of $\widehat{\Omega}$ is found, we reverse this sequence of transforms to place it back in physical space.

7.4 MOF Algorithms Details

7.4.1 Non-linear Optimization

We construct our objective function from the non-linear least-squares error in raw moments up to third order between the given reference moments and the actual moments of the reconstructed



(a) Original (b) Rotated (c) Scaled

Figure 7.2: Example of orientation-aligned material normalization. Given the original material (a), we apply a rotation (b) and scaling (c) to the space so that optimization can be performed in a reference space for which the reference ellipsoid (red) for the true material (gray) is spherical.

approximation to the material, but do so in reference space. In 3D, there are a total of 20 moments up to third order, which permits optimization over the 12 parameters of MOF^3_{4hs}, 15 parameters of MOF^3_{5hs}, or 18 parameters of MOF^3_{6hs} before the system becomes underdetermined. Each individual oriented plane $\widehat{\mathbf{n}} \cdot \mathbf{x} = d$ is parameterized by angles $\theta \in [0, 2\pi), \phi \in [0, \pi)$ that define normal $\widehat{\mathbf{n}} = (\cos(\theta)\sin(\phi), \sin(\theta)\sin(\phi), \cos(\phi))$, and the signed distance from the origin d. This provides us with the following objective function:

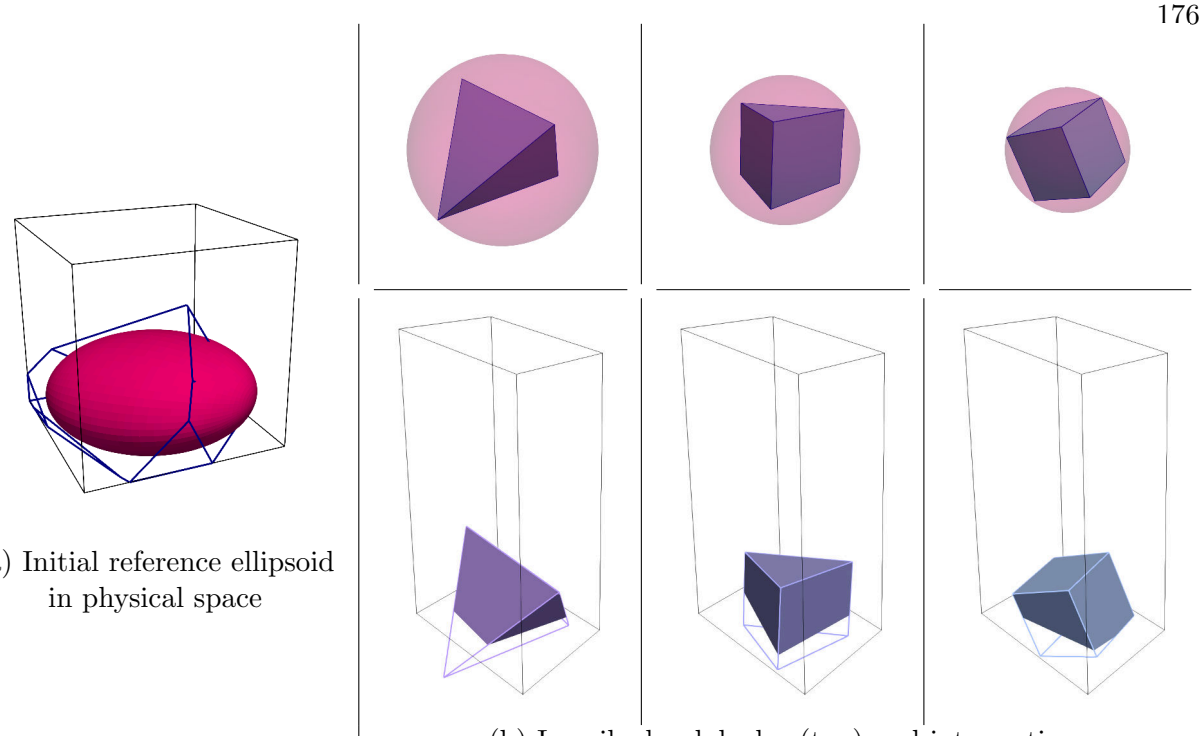
$$E(\Omega_r; \Omega) = \sum_{0 \le i+j+k \le 3} \left(\widehat{M}^r_{ijk} - \widehat{M}_{ijk} \right)^2 \tag{7.11}$$

with $\widehat{M}^r_{ijk}$ representing the raw moments calculated for the reconstruction and $\widehat{M}_{ijk}$ representing the reference moments scaled to the reference space. While we assume that reference moments will be provided via an outside application (i.e., hydrodynamic simulation), in this work we calculate reference moments for static tests from the intersection of high-polygon surface models and the cells of a mesh for demonstration purposes in this work. The computational cell C is also transformed into the same reference space, which we denote $\widehat{C}$.

We minimize this objective function using the Levenberg-Marquardt algorithm for solving unconstrained non-linear least-squares problems [107]. This algorithm is implemented in the C++ linear algebra library Eigen [57], which computes gradients of the objective function using centered finite differences in leiu of an analytic representation. In general, the dampening term used in the Levenberg-Marquardt algorithm makes it quite robust to an ill-conditioned Jacobian relative to standard Gauss-Newton solvers, making it particularly well-suited to this problem where only approximate gradients are available. The use of finite differences also means that the computational cost of optimization largely scales with the number of input parameters, as material moments up to third-order must be re-computed for changes in each. Doing so at each iteration, combined with the number of iterations necessary for convergence, requires the calculation of moments up to third-order be performed efficiently. For this reason we use the Interface Reconstruction Library (IRL), which provides routines for performing intersections between planes and general polyhedra using efficient half-edge data structures [30]. Fortunately, once the material polyhedron is computed by clipping the computational cell with each plane in the reconstruction, the cost of computing additional high-order moments is marginal.

The primary feature of the proposed method is to initialize this optimization procedure using the planes generated by inscribing a polyhedron in the reference ellipsoid, which is a unit sphere in reference space by construction. For MOF^3_{4hs} and MOF^3_{6hs} we use a single regular tetrahedron and cube respectively for this purpose, and for MOF^3_{5hs} we use a triangular prism. We show an example of each kind of inscribed polyhedron in Figure 7.3. The exact orientation of the initial polyhedron in the sphere is somewhat arbitrary, as the nature of the orientation-normalization means that no specific configuration is more likely to converge to the global minimum than any other. In the provided numerical experiments, we initialize the relevant polyhedron with the vertices in Table 7.1, defined in spherical coordinates.

Although the normalization scheme is also designed so that the reference ellipsoid (a sphere) has volume equal to the volume of the reference material, the inscribed polyhedron will naturally have a significantly smaller volume. As a result, we modify this shape prior to the initiation of



(a) Initial reference ellipsoid in physical space

(b) Inscribed polyhedra (top) and intersection with computational cell in reference space (bottom)

Figure 7.3: Example of initial inscribed polyhedra. We normalize reference moments so that the reference ellipsoid in physical space (a) is spherical in reference space. We then inscribe a single polyhedron with the corresponding number of faces (b) as the initial guess for optimization. Note that each sphere has been scaled in reference space so that the volume of the intersection of the inscribed shape with the cell is equal to the reference value.

the Levenberg-Marquardt algorithm so that the intersection of the inscribed polyhedron and $\widehat{C}$ has volume equal to that of the reference material $\widehat{M}^r_{000}$. This is an important preprocessing step for this specific implementation of Levenberg-Marquardt, which uses the initial error in each individual moment to define internal scaling factors. When this volume correction is not performed, the large difference in volume between the reference and approximate interface relative to the errors in first- and second- order moments (which are initially close to zero by construction) causes the algorithm to become unstable.

Tetrahedron	Triangular Prism	Cube
$(\pi/2, \arctan(\sqrt{2}/2))$	$(0, \arctan(\sqrt{2}))$	$(0, \arctan(\sqrt{2}))$
$(\pi/2, \pi - \arctan(\sqrt{2}/2))$	$(2\pi/3, \arctan(\sqrt{2}))$	$(\pi/2, \arctan(\sqrt{2}))$
$(\pi - \arctan(\sqrt{2}/2), 3\pi/2)$	$(4\pi/3, \arctan(\sqrt{2}))$	$(\pi, \arctan(\sqrt{2}))$
$(\arctan(\sqrt{2}/2), 3\pi/2)$	$(0, \pi - \arctan(\sqrt{2}))$	$(3\pi/2, \arctan(\sqrt{2}))$
	$(2\pi/3, \pi - \arctan(\sqrt{2}))$	$(0, \pi - \arctan(\sqrt{2}))$
	$(4\pi/3, \pi - \arctan(\sqrt{2}))$	$(\pi/2, \pi - \arctan(\sqrt{2}))$
		$(\pi, \pi - \arctan(\sqrt{2}))$
		$(3\pi/2, \pi - \arctan(\sqrt{2}))$

Table 7.1: Spherical coordinates (u, v) for each initial polyhedron inscribed in a unit sphere parameterized by $S(u, v) = (\cos(u)\sin(v), \sin(u)\sin(v), \cos(v))$.

7.4.1.1 Customized Levenberg-Marquardt Algorithm

While the standard unconstrained Levenberg-Marquardt solver using the default parameters supplied by Eigen is usually sufficient, we empirically observe improved convergence to a global solution in numerical tests when we impose geometric constraints on the involved planes. For example, we restrict the angle parameters of each plane, θ, ϕ, to the intervals $[0, 2\pi)$ and $[0, \pi)$ respectively using the modulo operator after each step of optimization to avoid numerical blowup that can occur when computed gradients become unstable, as well as restricting the maximum step size used during the calculation of finite differences for such periodic parameters.

More impactful is our restriction of the distance parameter d for each plane. Ordinarily, it is possible for a plane to become entirely disconnected from the computational cell. When this happens, perturbations in the planes' parameters do not result in any changes to the computed moments of the interface, and so all local gradients are zero. This means that planes become "stuck" when they are disconnected from the surface, and do not influence the material interface for all subsequent optimization iterations. As a result, we restrict the parameter d for each plane to an interval that ensures the plane intersects the computational cell.

To construct this interval for a given plane with unit normal $\hat{\mathbf{n}}$, we note the plane defined by $\hat{\mathbf{n}} \cdot \mathbf{x} = \hat{\mathbf{n}} \cdot \mathbf{v} = d$ necessarily contains the point $\mathbf{v}$, and moves continuously with d along the normal direction $\hat{\mathbf{n}}$. Because the computational cell is polyhedral, we need only check its vertices $\mathbf{v}$ to find the maximum and minimum values of $d = \hat{\mathbf{n}} \cdot \mathbf{v}$ which still correspond to planes $\hat{\mathbf{n}} \cdot \mathbf{x} = d$ that intersect the cell at least one point of the cell. Altogether, this mean that for each plane, we must clamp the value of the distance d to the interval

$$d \in [\min\{\hat{\mathbf{n}} \cdot \mathbf{v}\}, \max\{\hat{\mathbf{n}} \cdot \mathbf{v}\}]$$

after each optimization step to ensure the resulting plane still intersects the computational cell. While this step does not completely remove the possibility of the converged-to interface involving fewer planes than are parameterized by the method, it does improve the overall stability and robustness of the optimization algorithm.

7.4.2 Enforcing the Volume Constraint

It is necessary that the volume of the reconstructed interface Ω_r exactly matches that of the reference material Ω in order for other applications of interest utilizing the reconstruction to be properly volume conserving. While the specific implementation of Levenberg-Marquardt in Eigen solves the unconstrained non-linear least squares problem without accounting for this restriction on the volume, we observe empirically that the volume of the reconstruction immediately produced is nevertheless very close to the reference volume. To correct this further, we follow the procedure in [31], which contracts or dilates all planes of the reconstruction simultaneously along their normal direction to directly enforce the volume constraint. Specifically, we begin with an initial set of distances $\{d_i\}_{i=1}^N$ for each of N planes in the reconstruction, and find the scalar displacement δ for which the convex intersection of the cell and the same planes with distances $\{d_i + \delta\}_{i=1}^N$ has volume equal to the reference. Effectively, this becomes a non-linear root-finding problem that is solved with the Illinois method [38], a modified version of the Regula-Falsi method that is well suited for this application, as the non-linear function is monotonic, and an initial bracketing interval is

readily available by considering the vertices of the cell (See Figure 7.4.) We note that this necessarily introduces very small deviations in the remaining moments, but this effect is ultimately negligible to the overall quality of the reconstruction. It is for this reason that we do not apply the same constraint at every step in the optimization, as to do so would be unnecessarily computationally expensive.

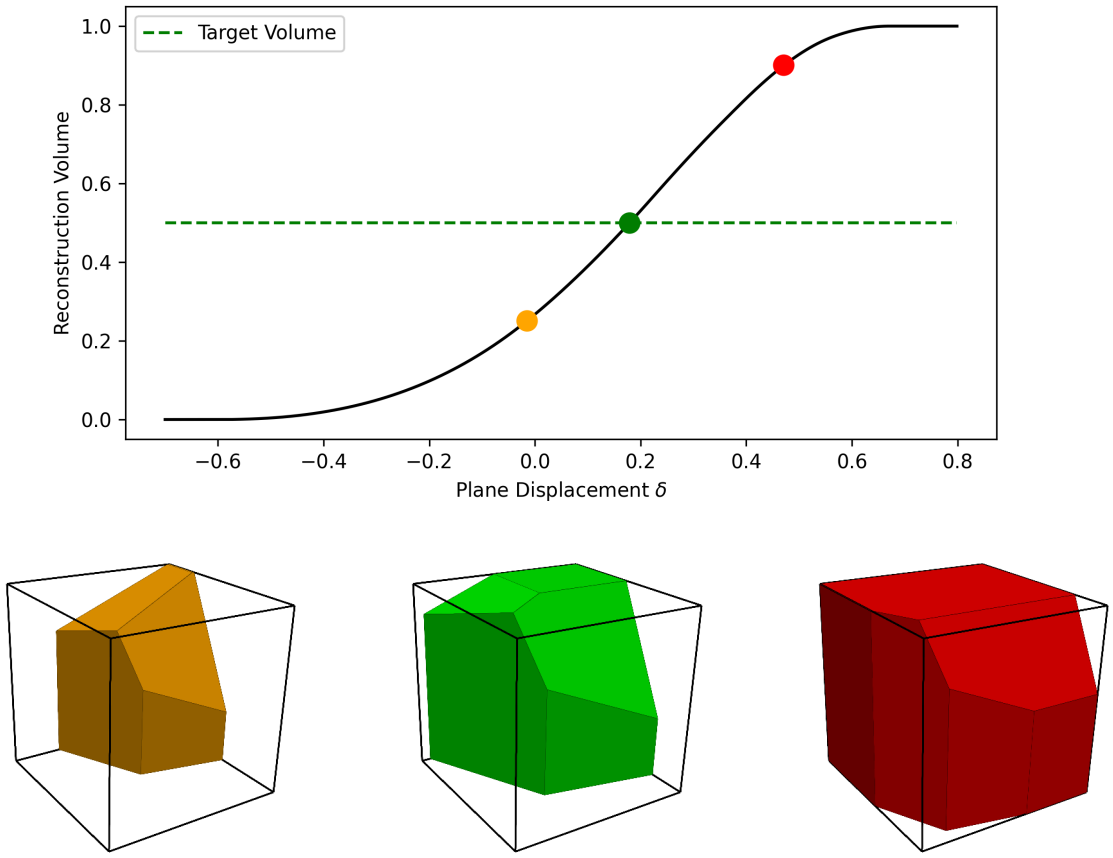


Figure 7.4: Example of root-finding problem for volume conservation. We plot the volume of captured material as a function of plane displacement parameter δ, and show the corresponding interface for different values of δ. We use the Illinois method to solve for the value of δ which gives the target volume, as the volume varies monotonically with the δ, and an initial bracketing interval is simple to compute.

7.4.3 Nonconvex Reconstruction

In this work, we have enforced that the material be equal to the convex intersection of the reconstruction planes, primarily for reasons of simplicity and efficiency in implementation. However, this can clearly only describe a subset of possible shapes. We extend this somewhat by allowing the half-spaces to represent the boundary of the *complement* of the convex intersection as well. While this too is only a second subset of possible shapes, their union is often sufficient for practical applications.

To the best of authors knowledge, there is no way of knowing *a priori* whether the underlying material is convex or not. To address this, we calculate two solutions for the non-linear optimization problem in each cell, one that assumes a convex reconstruction and one that assumes a reconstruction whose complement is convex. The first is performed using the original reference data M_{ijk}, while the second uses their complement in the unnormalized cell C, given by

$$M^c_{ijk} = M^C_{ijk} - M_{ijk},$$

where M^C_{ijk} are the raw moments of the unnormalized cell C. We note that these two reconstructions could be performed in parallel.

The results of each optimization are stored and then compared. Importantly, the proposed method uses a different normalization for each set of moment data, and so it is necessary for the compared errors to each be considered in the original, unnormalized, physical space. If it is found that the convex-complement method more accurately captures the reference moment data, then an internal flag is set to denote for downstream applications that utilize the reconstruction. Often, such applications will require decomposition of the non-convex material into a union of convex shapes, which we demonstrate in Figure 7.10.

7.5 Summary of the MOF Algorithm

We summarize the main steps of the proposed family of MOF interface reconstruction algorithms below. Given a computational cell, its reference moments, and the number of half-spaces N

used in the MOF^3_{Nhs} reconstruction, perform the following steps:

(1) Normalize the reference moments so that the reference ellipsoid is spherical (Section 7.3.2)

(2) Construct an initial polyhedron inscribed in the sphere (Table 7.1)

(3) Enforce that the polyhedron has the same volume in the cell as the normalized reference data (Section 7.4.2)

(4) Apply the Levenberg-Marquardt algorithm on the planes defined by the polyhedron (Section 7.4.1)

(5) Again, enforce that the reconstruction has the same volume in the cell as the normalized reference data (Section 7.4.2)

(6) Repeat the above steps for the complement of the reference moment data (Section 7.4.3)

(7) Evaluate the error for each reconstruction against their respective reference moment data:

- If the first error is smaller, return the intersection of the half-spaces in the first reconstruction.
- If the second error is smaller, return the *complement* of the intersection of the half-spaces in the second reconstruction as the union of a set of convex polyhedra.

7.6 Results

7.6.1 Single Cell Tests

We begin by demonstrating this method on simple single-cell examples for known shapes where an exact reconstruction is possible. In such cases where the target shape is known, we can use an error metric derived from the *symmetric difference*, which is defined for arbitrary sets as the total region contained in either set, but not their intersection. In the context of material interface reconstruction, this symmetric difference error metric measures the sum of the volume of the true

material that our approximation has failed to capture, and the volume of the approximation that does not capture the true material. Written mathematically, we have

$$S(\Omega_r;\Omega) := \int_{\Omega_r\setminus\Omega} dV + \int_{\Omega\setminus\Omega_r} dV. \tag{7.12}$$

In practice, however, we only have access to the moment-derived least-squares error as an error metric for the reconstruction $E(\Omega_r;\Omega)$ (see Section 7.4.1).

7.6.1.1 Exact Reconstructions

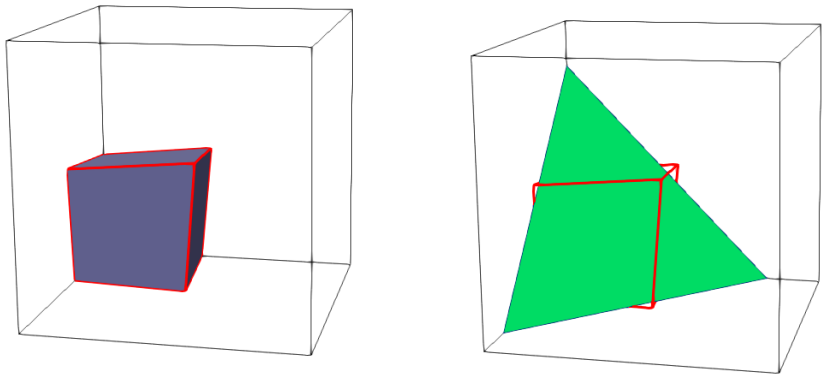
We show in Figure 7.5 the "vertex-corner" shape introduced in Section 7.2. The computational cell is taken to be the cube $[-0.5, 0.5]^3$, with the three planes with normals and distances defined by

$$\widehat{n}_1 = \langle 1.0, -0.1, -0.1\rangle, \quad d_1 = 0.0,$$

$$\widehat{n}_2 = \langle -0.1, 1.0, -0.1\rangle, \quad d_2 = 0.0,$$

$$\widehat{n}_3 = \langle -0.1, -0.1, 1.0\rangle, \quad d_3 = 0.0.$$

As expected, MOF^3_{4hs} is able to perfectly recreate this shape, both exactly matching each of the 20 reference moments and resulting in zero symmetric difference error, shown in Table 7.2. We note that although MOF^3_{4hs} is in some sense over-parameterized for this geometry, as we allow the placement of four planes when three is sufficient to achieve an exact reconstruction, we are able to exactly reconstruct the shape. In this example, the redundant plane is tangent to the resulting material polyhedron at its optimum placement, but does not influence its shape. Similar results are observed with MOF^3_{5hs} and MOF^3_{6hs}, but with more redundant planes in the final reconstruction. For an additional point of comparison, we make reference to a simple one-plane MOF method utilizing third-order moments. We initialize this MOF^3_{1hs} method with a plane whose normal points from the material centroid to the cell centroid, a reasonably effective initial guess for such a method. As shown in Figure 7.5, this provides a poor approximation of the geometry, and will perform even worse in following examples, where even more portions of the shape are detached from the cell boundary.



(a) True material (red) and the reconstruction from moments

(b) One-plane reconstruction

Figure 7.5: A comparison between the proposed multi-plane MOF method (a) and a single-plane MOF method (b) on a corner feature. We exactly reconstruct the "vertex-corner" using three of the four planes used in MOF^3_{4hs}.

We perform a similar test on the same computational cell for the "tip" shape in Figure 7.6, defined by the planes

$$\begin{aligned}
\widehat{n}_1 &= (-1.0, 0.0, 0.2)\,, \quad d_1 = 0.1, \\
\widehat{n}_2 &= \left(\sqrt{2}, \sqrt{2}, 0.2\right), \quad d_2 = 0.0, \\
\widehat{n}_3 &= \left(\sqrt{2}, \sqrt{2}, 0.2\right), \quad d_3 = 0.0.
\end{aligned}$$

Again, MOF^3_{4hs} achieves an exact reconstruction while a one-plane method necessarily makes the shape unrecognizable to maintain the volume constraint.

To further test the robustness of our method, we repeat this test in Figure 7.7 using the same "tip" shape, but occupying a much smaller volume in the cell. Specifically, we translate the cell along the positive z direction to occupy the $[-0.5, 0.5] \times [-0.5, 0.5] \times [0.2986, 1.2986]$ so that the section of filament that intersects with the cell has the same geometry, but a calculated volume of

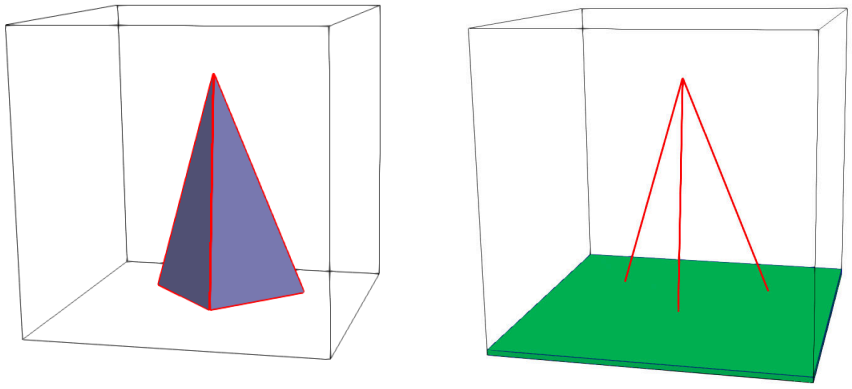
Moment	MOF^3_{1hs}	MOF^3_{4hs}
M_{000}	-9.57776×10^{-13}	-1.38778×10^{-17}
M_{100}	-0.00264148	6.93889×10^{-18}
M_{010}	-0.00293966	1.73472×10^{-17}
M_{001}	-0.00232491	1.38778×10^{-17}
M_{200}	0.00218141	-1.73472×10^{-18}
M_{110}	0.000683377	-5.20417×10^{-18}
M_{101}	0.000475685	-3.46945×10^{-18}
M_{020}	0.00228779	-8.67362×10^{-18}
M_{011}	0.000575576	-5.20417×10^{-18}
M_{002}	0.00207197	-6.93889×10^{-18}
M_{300}	-0.00107547	-2.1684×10^{-18}
M_{210}	-0.000503406	8.67362×10^{-19}
M_{201}	-0.000422462	0
M_{102}	-0.0004949	1.30104×10^{-18}
M_{111}	0.000259171	1.30104×10^{-18}
M_{102}	-0.00043277	8.67362×10^{-19}
M_{030}	-0.00112707	-4.33681×10^{-19}
M_{021}	-0.000452824	1.30104×10^{-18}
M_{012}	-0.000471645	8.67362×10^{-19}
M_{003}	-0.00102168	0
$E(\Omega_r; \Omega)$	0.0574	1.923×10^{-17}
Symmetric Difference	0.645554	2.498×10^{-15}

Table 7.2: Per-moment errors between reference data and the reconstructed interface for the "vertex-corner" example in Figure 7.5.

3.19117×10^{-14}. The resulting tetrahedron has coordinates at

$$\begin{aligned} p_1 &= (-0.04226039, 3.1973229 \times 10^{-5}, 0.298600) \\ p_2 &= (-0.04226039, -3.1973229 \times 10^{-5}, 0.298600) \\ p_3 &= (-0.042228417, 7.6165203 \times 10^{-18}, 0.298600) \\ p_4 &= (-0.042241661, 7.609744 \times 10^{-18}, 0.29869365) \end{aligned} \tag{7.13}$$

Despite needing to handle a volume that is close to machine precision, our family of methods is nevertheless able to match the shape exactly. This is largely due to our unique normalization procedure, which causes the method to be extremely robust to the size of the material. As a result of this normalization, the optimization performed previous example and this one are near-identical

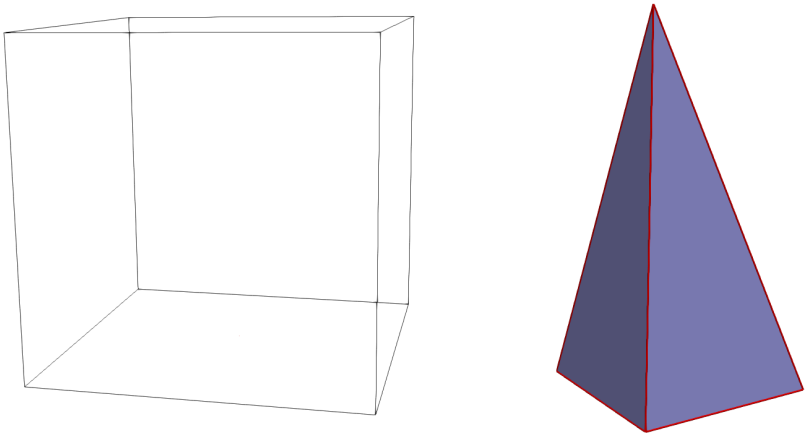


(a) True material (red) and the reconstruction from moments

(b) One-plane reconstruction

Figure 7.6: A comparison between the proposed multi-plane MOF method (a) and a single-plane MOF method (b) on a "tip" feature. We exactly reconstruct the shape using three of the four planes in MOF^3_{4hs}.

in reference space, with the only difference being a larger normalized cell for the smaller material.



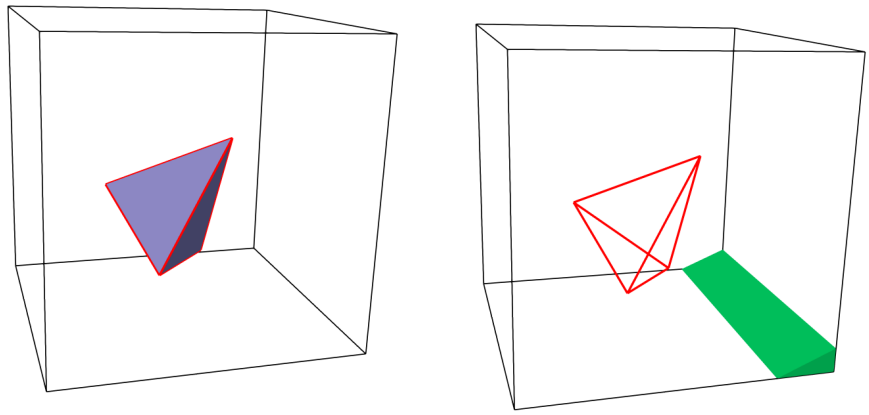
(a) True material (red) and the reconstruction from moments on a $[-0.5, 0.5]^3$ computational cell.

(b) Figure zoomed to material. $E(\Omega_r; \Omega) = 1.874 \times 10^{-16}$ $S(\Omega_r; \Omega) = 1.425 \times 10^{-17}$

Figure 7.7: Robustness test for small material volume. Note that relative to the cell, the true material (a) is so small as to occupy only a single pixel in this rendering. Nevertheless, the proposed method achieves an exact reconstruction (b).

Finally, we consider in Figure 7.8 an example that is even more adversarial for MOF^3_{1hs}, in which the material is entirely disconnected from the cell boundary. As before, the proposed

MOF^3_{4hs} method is able to accurately recover the shape, while having, in some sense, the mximum error possible with respect to the symmetric difference.



(a) True material (red) and the reconstruction from moments

(b) One-plane reconstruction

Figure 7.8: A comparison between the proposed multi-plane MOF method (a) and a single-plane MOF method (b) on material that is fully embedded within the computational cell. We exactly reconstruct the shape using all four planes in MOF^3_{4hs}.

7.6.1.2 Inexact Reconstructions

In addition to the sharp target features depicted in Figure 7.1, our method is also capable of approximating shapes with smooth features, which no planar reconstruction can recover exactly. We perform these tests using the MOF^3_{6hs} method as a representative of the proposed family.

In the following examples, we consider the intersection of our computational cell with ellipsoids of varying size, position, and orientation. To obtain the reference moments, we construct a polyhedron that approximates the geometry of the curved object, from which we can compute the moments of its intersection with the cell using the clipping methods from the IRL software package described in Section 7.4.1. While these do not exactly match the moments of a true truncated ellipsoid, the symmetric difference errors are computed with respect to this approximating polyhedron, and not the ground truth geometry.

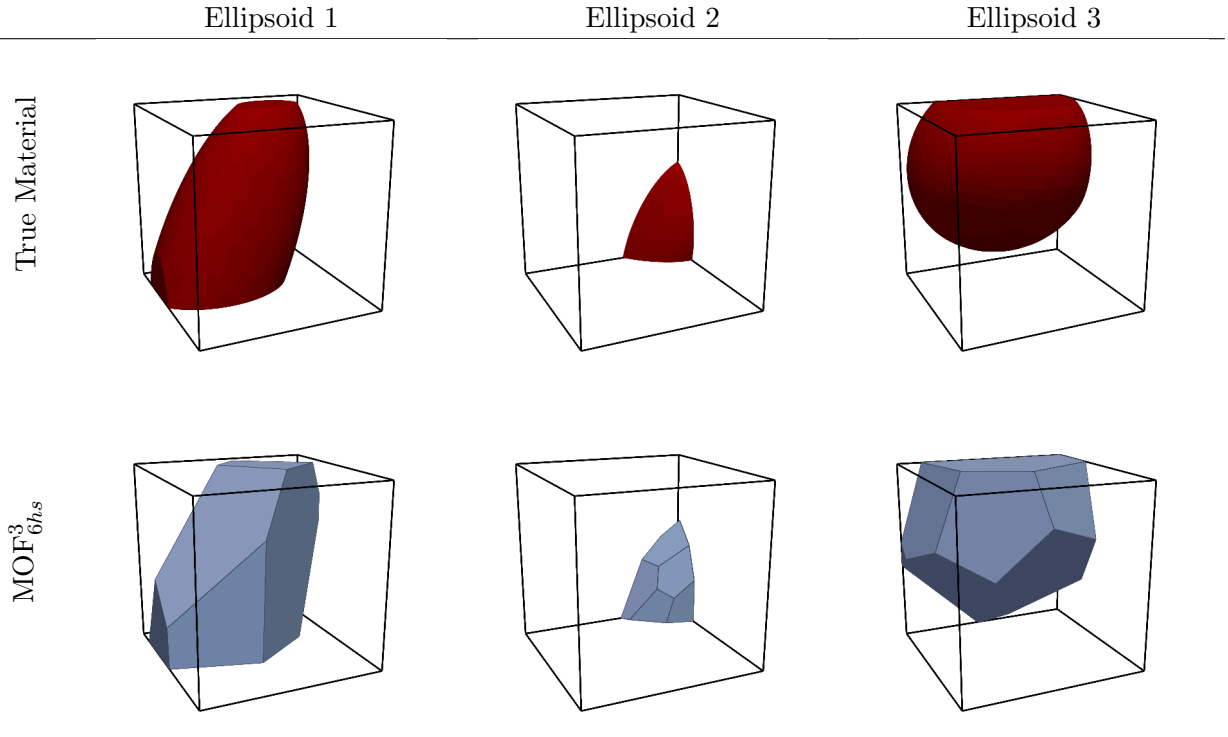
In Figure 7.9, we consider three such ellipsoids, and define each for reproducibility with

	a	b	c	α	β	γ	x_0	y_0	z_0
Ellipsoid 1	0.32	0.40	0.98	4.9	-1.4	3.3	-0.10	0.34	0.20
Ellipsoid 2	0.35	0.23	0.58	5.1	2.3	4.0	0.37	0.41	0.59
Ellipsoid 3	0.48	0.60	0.57	4.7	4.0	1.7	0.10	-0.18	0.48

Table 7.3: Parameters for single-cell ellipsoid tests in Figure 7.9.

parameters defined by the values in Table 7.3. Numerically, we parameterize these ellipsoids through the lengths of their three axes a, b, c (parallel to the x-, y-, and z-axis respectively) and the rotation matrix $R(\alpha, \beta, \gamma)$ defined in Equation 7.5, followed by a shift to have the center (x_0, y_0, z_0).

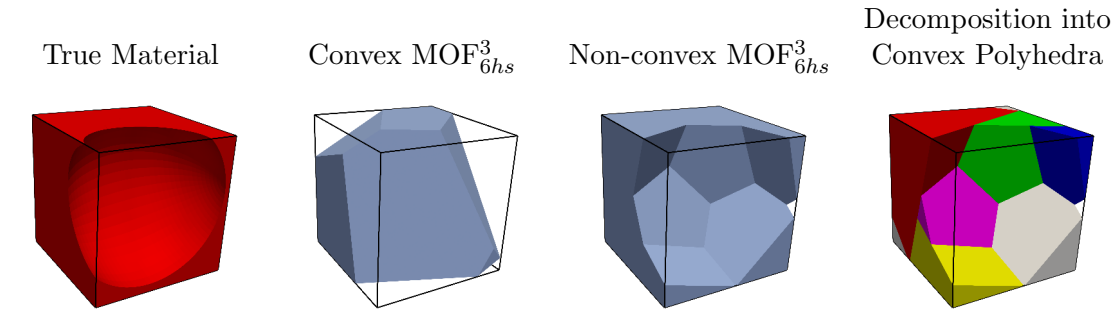
As seen in Figure 7.9, we see qualitative agreement with the ground truth material, as well as error metrics which indicate that we have accurately represented the surface by capturing the relevant geometric features.



	Ellipsoid 1	Ellipsoid 2	Ellipsoid 3
$E(\Omega_r)$	0.0004765	1.04518×10^{-6}	0.0002632
$S(\Omega_r)$	0.042901	0.0005620	0.0244484

Figure 7.9: Example reconstructions on curved shapes. We test MOF^3_{6hs} on shapes for which an exact reconstruction is not possible. Despite this, the reconstructed interface accurately captures important geometric features.

We can also test our method on the *complement* of the third ellipsoid to demonstrate the ability of the proposed methods to reproduce non-convex shapes. We plot in Figure 7.10 both the trial convex reconstruction and convex-complement reconstruction, observing that because the convex-complement reconstruction has a lower moment error, it is accepted as the final reconstruction. Importantly, because the moment data over which the optimization is performed is identical between this case and that demonstrated for Ellipsoid 3 in Figure 7.9, the material approximations returned by the two methods are themselves complements. It is for this reason that we primarily restrict our numerical tests to convex material, as the problem of reconstructing materials whose compliment is convex in the cell is *equivalent* to the problem of reconstruction convex material.



$E(\Omega_r)$	0.0197032	0.000237875
$S(\Omega_r)$	0.190205	0.0233722

Figure 7.10: Example of reconstruction of non-convex material. Because the interface generated by the complement of a convex shape produces lower moment error, it is returned as the final interface. We also demonstrate the decomposition of a reconstructed non-convex material into convex polyhedra for related applications utilizing the reconstruction.

Finally, we can also use smooth shapes to illustrate differences between the MOF$^3_{4hs}$, MOF$^3_{5hs}$, and MOF$^3_{6hs}$ methods. In cases where an exact reconstruction is possible, the trade-off of interest is that optimizing over additional (potentially redundant) planes improves the *reliability* of the method, in other words the likelihood of an accurate recovery from a single initial condition. However, when exact recovery is not possible, the relevant trade-off is instead between computational cost and *accuracy*, as more planes in the reconstruction mean that a better approximation of curved geometry is possible. As we can see in Figure 7.11, increasing the number of planes in the recon-

struction does improve the error, but also the total cost of the method. Importantly, while the cost of each method scales directly with the number of planes in the reconstruction, the complexity of the polyhedral cell also has a large impact on the performance of the method, as additional cell faces each have the same performance impact on the moment calculation as an additional reconstruction plane.

In spite of this example, we note that it is not necessarily the case that a reconstruction with more planes will *always* result in greater reconstruction accuracy. For example, we observe for Ellipsoid 2 of Figure 7.9 that the optimal interface found by our method is defined by five planes, rather than the six that the global minimum foudn by the MOF^3_{6hs} method would be expected to have. In principle, this is because the proposed family of multi-plane methods are designed to reliably converge to accurate *three*-plane solutions, which is the maximum number necessary to exactly capture the sharp features in Figure 7.1. On the other hand, shapes with only flat features are well-approximated by fewer planes, which introduces spurious local minima for the optimization problem. From this perspective, it is in some sense an unintended advantage of our reconstruction approach that for smooth shapes with sharp features, it tends to converge to solutions that utilize as many planes as are parameterized (We explore this further in the discussion of Figure 7.18).

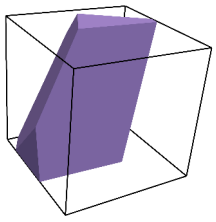
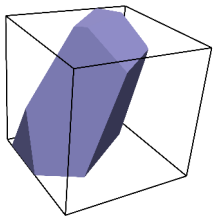
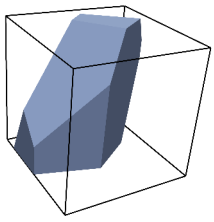
	MOF^3_{4hs}	MOF^3_{5hs}	MOF^3_{6hs}
Reconstructed Ellipsoid			
$E(\Omega_r)$	0.00188321	0.00102675	0.000476561
$S(\Omega_r)$	0.0675061	0.0525182	0.0429109
Total Time (s)	0.0805	0.174	0.203
Iteration Count	20	33	21
Time per Iteration (s)	0.004025	0.00527	0.00967

Figure 7.11: Comparison of run-time vs. accuracy among multi-plane MOF methods. In cases where an exact reconstruction is not possible, the use of additional planes results in an approximated interface that is more computationally expensive, but can result in considerably lower error.

7.6.2 Mesh Tests

As would be the case in a practical application, it is necessary that our method is performant when considered across *all* computational cells in a mesh. We note that the local nature of any moment-of-fluid method means that accuracy across an entire mesh be equivalent to accuracy across each individual cell. However, in addition to providing a wide breadth of single-cell examples, we evaluate the proposed method across an entire mesh to demonstrate the quantitative advantages of a multi-plane reconstruction scheme.

7.6.2.1 Cube Test

In our first example in Figure 7.12, we consider MOF^3_{1hs} and MOF^3_{4hs} on the cube $[-0.5, 0.5]^3$ that we have rotated by $R(1.2, -0.82, 1.0)$ around the origin (See Equation 7.5) and then translated by $(0.05, 0.10, -0.05)$ to avoid unfairly exploiting the shape's symmetry. This shape is then reconstructed on an $N \times N \times N$ uniform grid of cells across the larger cube $[-1, 1]^3$. Although the MOF^3_{1hs} method used for comparison does approach the ground truth with successive refinement, the nature of the shape's corner features makes it impossible to represent the shape exactly. Furthermore, the orientation of the shape makes it difficult for a PLIC method recover its edges exactly at any refinement level on an axis-aligned grid, even in the presence of a more sophisticated, adaptive refinement scheme.

In these examples, we consider in Table 7.4 the maximum of both the moment error $E(\Omega_r; \Omega)$ and the symmetric difference error $S(\Omega_r; \Omega)$ across all cells in the mesh. To account for potential outliers in the values of these error metrics across cells, we also consider their average across all mixed cells in the mesh, that is to say, cells which are neither empty nor completely filled by the material. In Figure 7.12 we see that even at the coarsest level of refinement, our MOF^3_{4hs} method visually recovers the shape near-identically, in contrast to the unrecognizable geometry produced MOF^3_{1hs} method. These results are supported numerically by Table 7.4, which shows that we indeed produce an interface with less error along each of the described metrics.

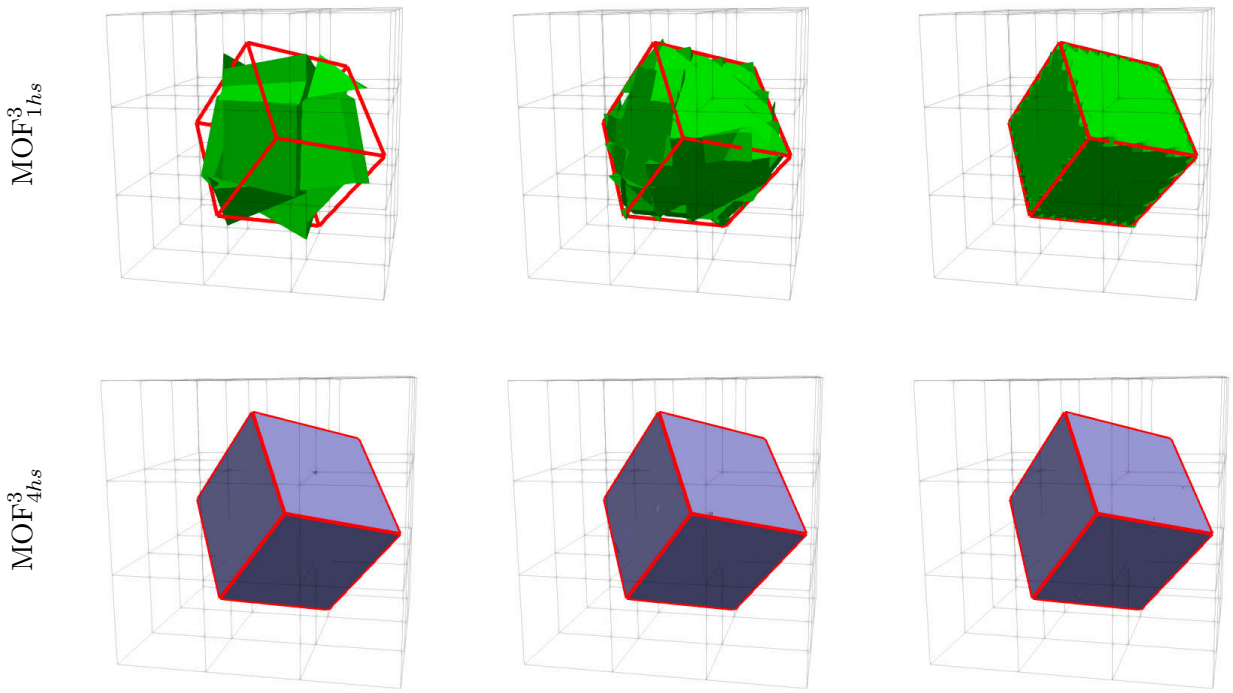


Figure 7.12: Comparison between single-plane and multi-plane reconstruction schemes for a cube across a Cartesian mesh. Using a multi-plane reconstruction scheme allows for more accurate reproductions on coarser grids. The true material is outlined in red. The relevant error metrics are presented in Table 7.4.

7.6.2.2 Ellipsoid Test

To consider an example that better resembles practical application, we apply MOF$^3_{4hs}$, MOF$^3_{5hs}$, and MOF$^3_{6hs}$ on a convex shape that cannot be recovered exactly in Figure 7.13, which depicts an ellipsoid defined by the parameters in Table 7.5. As in the previous example, we consider an orientation of the shape that is not aligned with the grid of cells, which are placed in the cube $[-1, 1]^3$.

We see in this example that the multi-plane reconstruction offered by each of MOF$^3_{4hs}$, MOF$^3_{5hs}$, and MOF$^3_{6hs}$ each offer a dramatic improvement over the single-plane technique, adequately recovering the primary geometric features even at remarkably low resolution grids. Even in the case of a single-cell grid, we can see that our method is able to recover a crude approximation

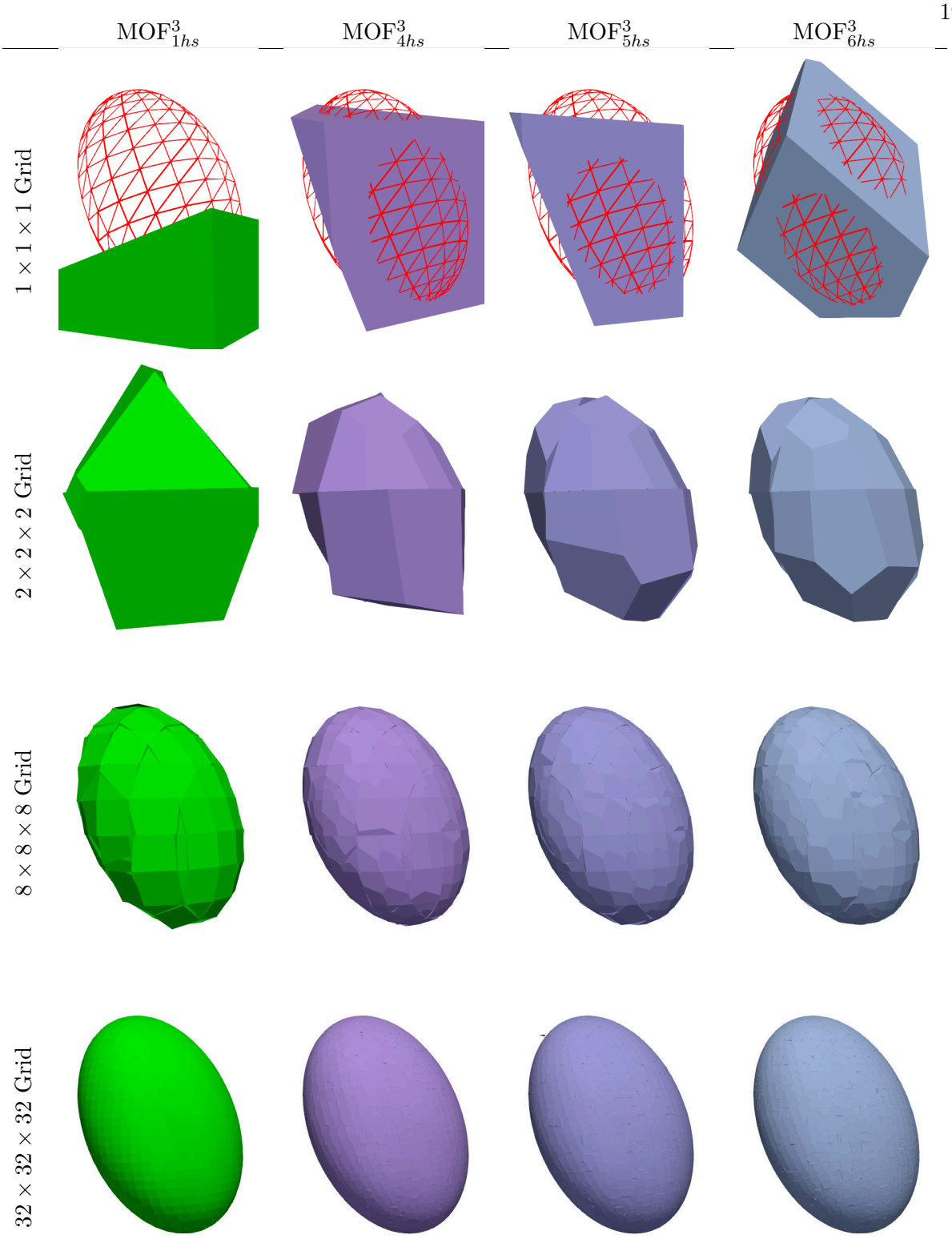


Figure 7.13: Comparison between single-plane and several multi-plane MOF reconstruction schemes for an ellipsoid over a Cartesian mesh. Using a multi-plane reconstruction scheme allows for more accurate reproductions on coarser grids. The target ellipsoid is suggested in red wireframe on the first row, although the shape from which ground truth moments are computed is of considerably higher resolution.

$E(\Omega_r;\Omega)$

Grid Resolution (# Mixed Cells)		$3\times3\times3$ (23)	$9\times9\times9$ (189)	$27\times27\times27$ (1752)
MOF^1_{1hs}	Cell Average	1.3284e-03	6.0892e-06	2.9583e-08
	Maximum	5.4000e-03	8.3435e-05	1.2974e-06
MOF^1_{4hs}	Cell Average	2.0053e-07	2.4080e-07	1.1596e-10
	Maximum	4.3996e-06	1.2727e-07	5.0307e-08

$S(\Omega_r;\Omega)$

Grid Resolution (# Mixed Cells)		$3\times3\times3$ (23)	$9\times9\times9$ (189)	$27\times27\times27$ (1752)
MOF^1_{1hs}	Cell Average	4.1915e-02	1.8975e-04	2.5371e-06
	Maximum	1.2922e-02	1.8596e-03	6.7378e-05
MOF^1_{4hs}	Cell Average	1.0440e-05	7.2676e-06	1.9463e-08
	Maximum	2.1353e-04	1.2985e-05	9.7976e-06

Table 7.4: Evaluation of our moment error metric $E(\Omega_r;\Omega)$ and the symmetric difference error metric $S(\Omega_r;\Omega)$ for the cube example in Figure 7.12. We see that the MOF^3_{4hs} method results in a significantly error across both metrics.

	a	b	c	α	β	γ	x_0	y_0	z_0
Ellipsoid	0.1	-0.1	0.1	0.9	-1.07	0.7	3.30719	0.18485	4.93365

Table 7.5: Parameters for multi-cell ellipsoid test in Figures 7.13, 7.14, and 7.15

of the completely embedded material, which is impossible when only a single plane is used in the reconstruction.

Additionally, considering reconstruction accuracy across an entire mesh allows for a more comprehensive analysis of how grid refinement impacts each of the proposed methods. As one would expect, Figure 7.13 demonstrates that the reconstruction is more accurate both with increasing levels of grid refinement and increasing numbers of planes used in the representation. However, we also observe that it is for the coarser grids that increasing the number of planes brings about the most improvement. For example, MOF^3_{6hs} offers a dramatic improvement over MOF^3_{1hs}, and even over MOF^3_{4hs}, on the $2\times2\times2$ grid, being clearly recognizable as depicting the original ellipsoid shape. On the other hand, the different methods visually perform very similarly on a $32\times32\times32$ grid. This is because as the cell-size decreases, the local geometry of the shape becomes increasingly

flat, and the POM in each cell is already well approximated by a single plane. We note, though, that this is not always the case, as shapes with sharp features (as in Figure 7.12) cannot be well approximated by a PLIC method at any level of refinement for an arbitrary mesh.

We further explore this point on the same ellipsoid shape by considering reconstruction accuracy as function of cell index through the heatmaps in Figure 7.14, where the two error metrics are evaluated over a $16 \times 16 \times 16$ grid. We note that although there is a considerable amount of cell-to-cell variability for each error metric within a single shape, this can be attributed to varying material volume in each cell. Nevertheless, cell-to-cell comparisons between the different methods are appropriate, as the interface in each cell's reconstruction is attempting to match the same data.

We observe in Figure 7.14(a) that the accuracy of the MOF^3_{1hs} method, both in terms of moment error $E(\Omega_r;\Omega)$ and the symmetric difference error $S(\Omega_r;\Omega)$, generally correlates with the curvature of the shape, where areas of high curvature tend to have higher reconstruction error. At the same time, the proposed multi-plane methods are far more robust to such qualities and, as seen in Figure 7.13, can be observed to outperform MOF^3_{1hs} most in regions of high curvature. We consider the overall distribution of these errors more directly through histograms over the same data in Figure 7.14(b), where we observe that we see a broad improvement in error across the domain of each of our methods against MOF^3_{1hs}.

At the same time, we observe through in this example that the difference between the MOF^3_{4hs}, MOF^3_{5hs}, and MOF^3_{6hs} methods is somewhat minimal. This is largely due to the resolution of the grid, as the curvature across a single cell within $16 \times 16 \times 16$ grid is (on average) not high enough to see strong visual improvement between the three methods. On the other hand, we can repeat this analysis for a coarser $4 \times 4 \times 4$ grid in Figure 7.15 and see more clearly the advantage of using MOF^3_{6hs} over MOF^3_{5hs} and MOF^3_{4hs}.

7.6.2.3 Nonconvex Test

Next, we consider the proposed family of methods on a shape with many non-convex features in the form of five "creases" positioned around an ellipsoid shape. We generate this shape according

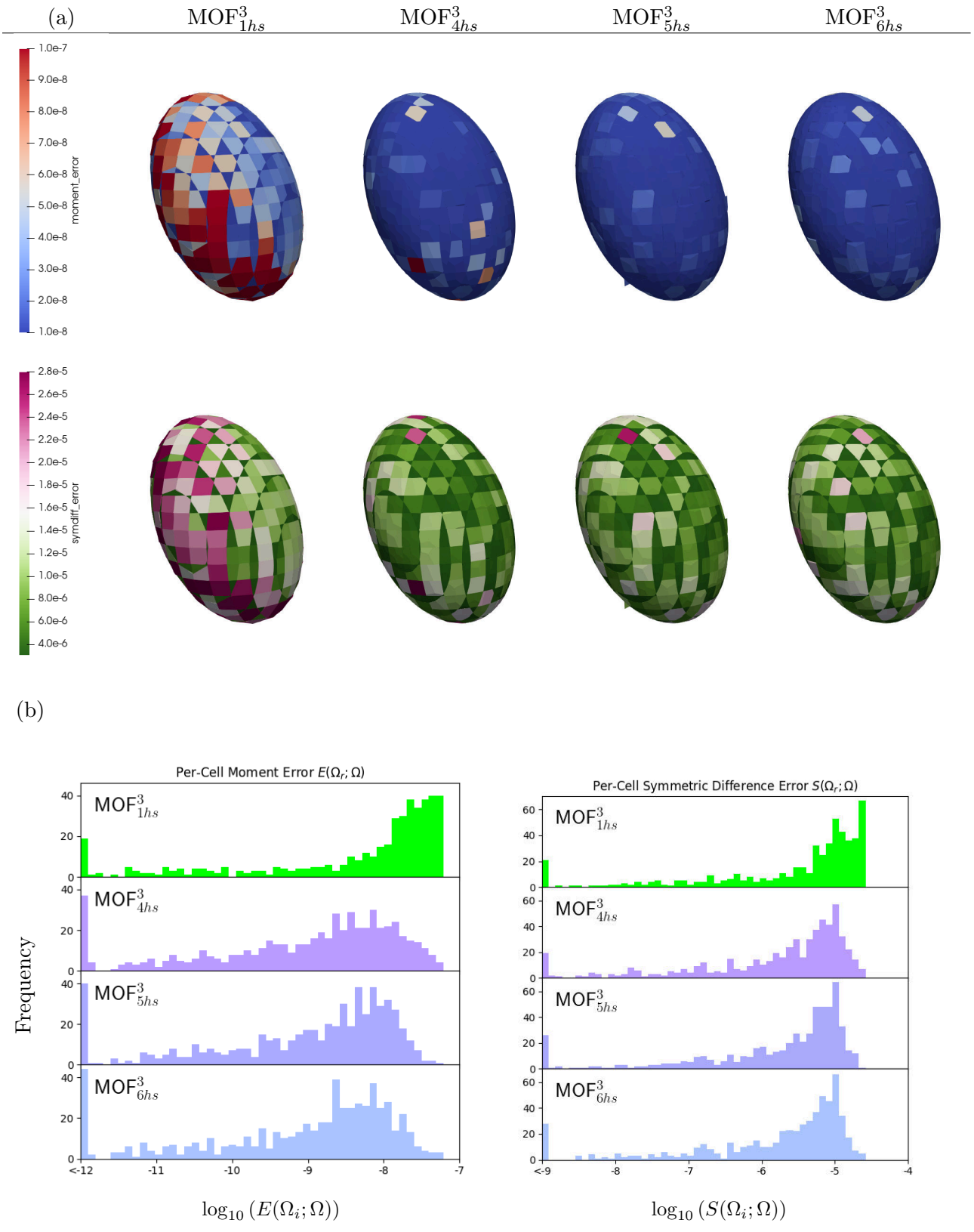


Figure 7.14: Error comparison between methods evaluated for an ellipsoid over a Cartesian mesh. Although MOF$^3_{1hs}$ can provide a sufficient representation for regions of low curvature, the use of a multi-plane scheme makes the accuracy of the reconstruction more tolerant to regions of higher curvature. Even still, the proposed family of multi-plane schemes still improves accuracy over one-plane alternatives when considered across the entire shape.

Per-Cell Errors on $4 \times 4 \times 4$ Mesh Grid

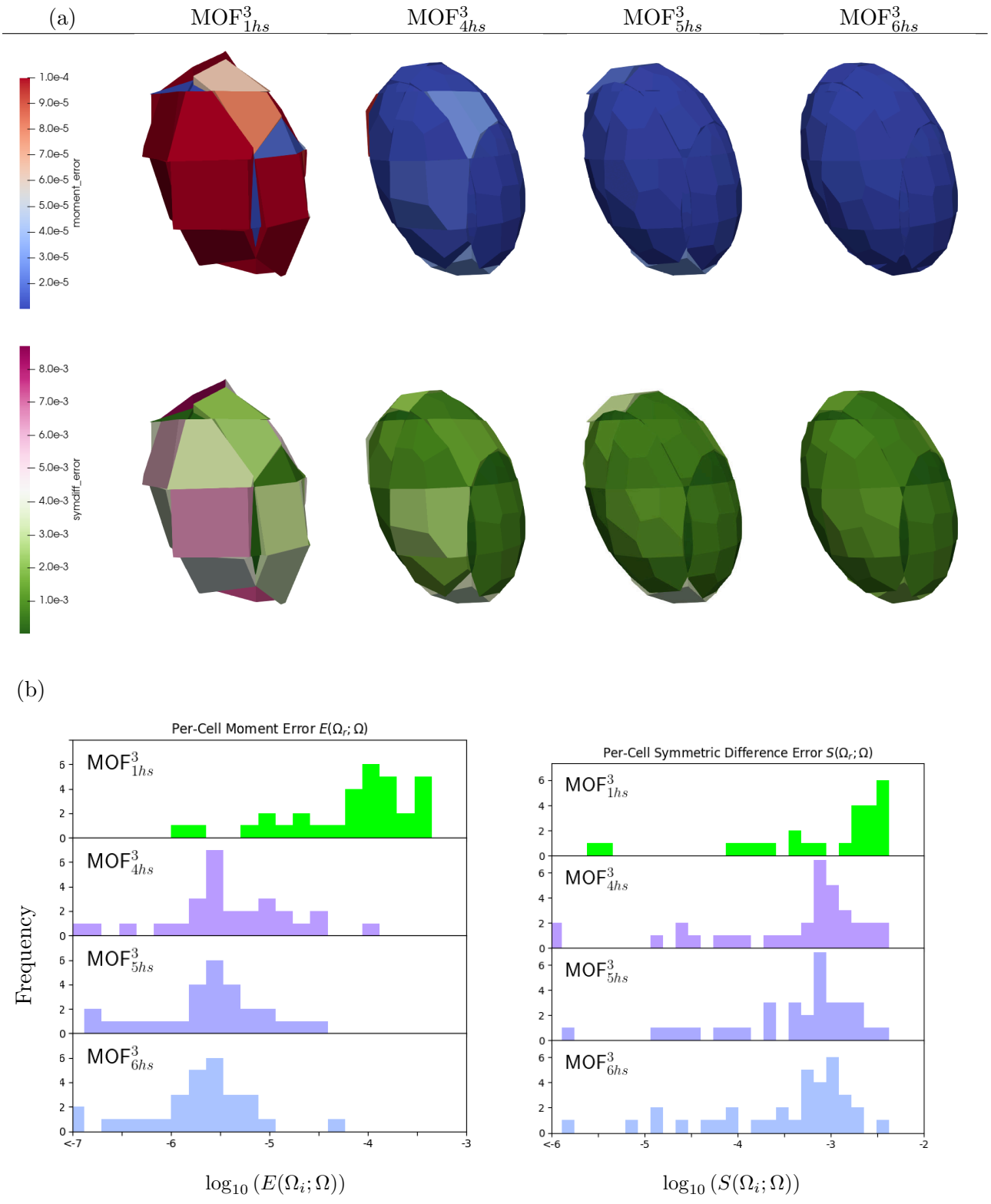


Figure 7.15: The results of Figure 7.14 reproduced on a coarser grid. We can see more clearly the advantages of using more planes per cell in the reconstruction, as MOF$^3_{5hs}$ and MOF$^3_{6hs}$ are shown to be significantly more performant than MOF$^3_{4hs}$.

to the following parametric surface:

$$x(u,v) = 0.04\left(1.5\sin(u)(6\cos(v) - \cos(6v)\right),$$

$$y(u,v) = 0.04\left(1.5\sin(u)(6\sin(v) - \sin(6v)\right),$$

$$z(u,v) = 0.24\cos(u).$$

The shape generated by this surface is then rotated around the origin by the rotation matrix $R(-0.1, -1.67, 0.0)$ and translated by the point $(0.2, 0.2, -0.2)$. We see as before in Figure 7.16 that proposed family of methods can far more accurately capture subgrid and curved features of the shape, and that successive refinement of the mesh improves the quality of this reconstruction relative to the ground truth.

It is through this example that we demonstrate the principle advantage of MOF^3_{4hs}, MOF^3_{5hs}, and MOF^3_{6hs}: the ability to achieve the same level of accuracy on a coarser grid. To do this, we explore the minimum level of mesh refinement needed for the MOF^3_{1hs} PLIC method to achieve the same overall level of accuracy as each of our multi-plane methods.

We first consider the shape in its entirety over a $5 \times 5 \times 5$ grid, chosen heuristically as the minimum level over which the MOF^3_{Nhs} methods reasonably approximate the curved features of the shape. For each grid, we sum the symmetric difference error across cells in the mesh. In contrast to the moment error, the sum of the symmetric difference error metric directly represents the total error in the shape. In Figure 7.17, we see similar performance among the proposed family of methods, which suggests that few individual cells in the $5 \times 5 \times 5$ mesh have prominent enough sharp features to fully exert all planes available to the reconstruction method. However, we can see that an $7 \times 7 \times 7$ grid of cells is necessary to reconstruct the shape with accuracy roughly equal to the MOF^3_{4hs}, and an $8 \times 8 \times 8$ grid is needed to surpass the accuracy of MOF^3_{5hs} and MOF^3_{6hs}. This correlates to a nearly three-fold increase in the number of mixed-material cells (58 vs. 163), serving in part to justify the increased cost of a multi-plane MOF method compared to the analogous PLIC procedure.

In many ways, we consider such a demonstration to be more effective than formal convergence

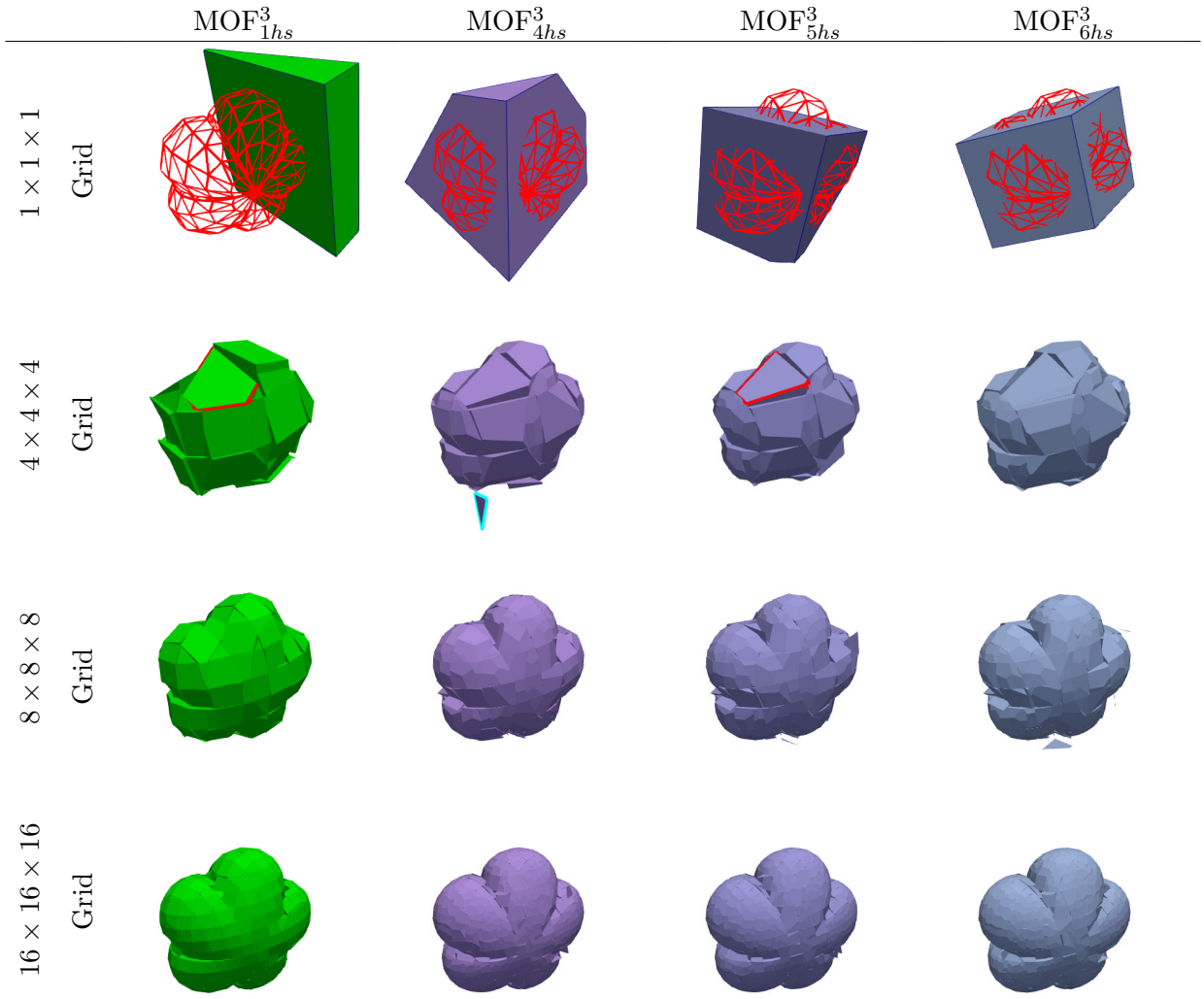


Figure 7.16: Comparison between reconstruction methods on a non-convex shape over a Cartesian mesh. Using a multi-plane reconstruction scheme permits reconstruction of non-convex regions wihin a single cell. The target shape is suggested in red wireframe on the first row, although the shape from which ground truth moments are computed is of considerably higher resolution. For the $4 \times 4 \times 4$ grid, we also emphasize in red the MOF$^3_{1hs}$ and MOF$^3_{5hs}$ reconstruction in one cell for further discussion in Section 7.7.1, and in cyan the MOF$^3_{4hs}$ reconstruction for further discussion in Section 7.7.2.

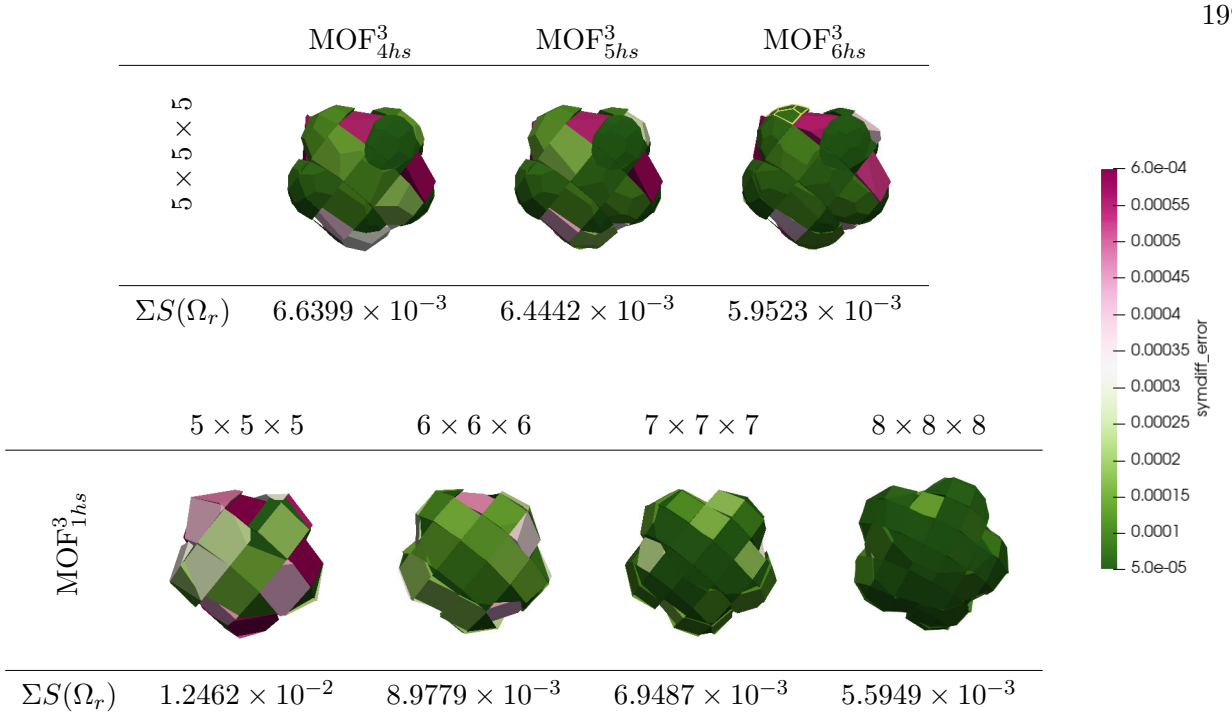


Figure 7.17: Error comparison across varying levels of refinement evaluated for a non-convex material. Additional levels of refinement are needed for a PLIC method to achieve the same aggregate levels of accuracy. In this example, an $8 \times 8 \times 8$ grid is required for MOF$^3_{1hs}$ to achieve the same total symmetric difference error as MOF$^3_{6hs}$. At the top of the MOF$^3_{6hs}$ reconstruction we have emphasized one POM in yellow for further discussion in Figure 7.18.

analysis for exploring the performance of our reconstruction technique, as the particulars of the ground truth geometry often have a far more significant influence on reconstruction accuracy than the resolution of the mesh. For example, we observe in Figure 7.17 that the error of each MOF$^3_{Nhs}$ method on the $5\times5\times5$ grid varies considerably with the complexity of the geometric features present in a given cell. This is also true for the PLIC method on the same grid, but as the mesh resolution is increased, the error becomes more uniformly distributed among the cells. This primarily illustrates that as the size and frequency of cells containing problematic geometric features decrease, the total error decreases as well. The same is also true of our MOF$^3_{Nhs}$ methods, but as seen in Figure 7.11, there are comparatively fewer advantages to using our multi-plane methods when the geometry in a cell is already locally smooth.

To see this more clearly, we consider another example of mesh refinement the POM in the

MOF^3_{6hs} example of Figure 7.17 which we have highlighted in yellow. This shape, despite having relatively high curvature, has no sharp features at the coarsest level of resolution. In Figure 7.18, we can see that there is over an order of magnitude of difference between the PLIC method and the six-plane reconstruction on the original cell. However, as the mesh resolution increases, the comparative improvement becomes markedly smaller. In some sense, this represents a shortcoming of our numerical scheme. Although our choice of initial condition can reliably recover sharp features, it is not particularly well-suited for a high-fidelity reconstruction of smooth features, as there exist spurious local minima around reconstructions that use fewer planes than are available. For example, we observe in this case that the MOF^3_{5hs} method actually outperforms the MOF^3_{6hs} method on the $2 \times 2 \times 2$ and $3 \times 3 \times 3$ grid, despite having fewer parameters. Ultimately, however, the key result of this figure is that the MOF^3_{6hs} method is able to reconstruct features on a single cell with an accuracy that the PLIC method requires a $3 \times 3 \times 3$ grid to achieve.

7.7 Discussion

7.7.1 Reconstruction of Nonconvex Material

The proposed family of methods are intended to accurately represent both convex material *or* material whose complement is convex, but many shapes have features for which neither is the case. This is shown somewhat adversarially in the example of Section 7.6.2.3, which has the property where all non-convex material localized to a cell is *also* not the complement of a convex shape. We highlight all such cells in Figure 7.19, and can observe that the error in the method along either error metric is concentrated to cells which do not contain convex material. Indeed, in all cells that *do* contain convex material, the MOF^3_{6hs} method achieves significant agreement with the ground-truth geometry.

We also observe through Figure 7.19 that the majority of cells use a convex material in the approximation, despite the material itself being non-convex. For a particularly difficult example, we refer back to the material highlighted in red in Figure 7.16, for which MOF^3_{5hs} uses a convex

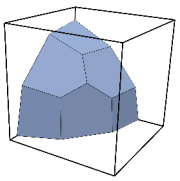
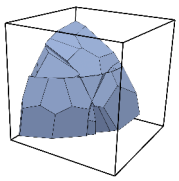
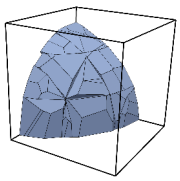
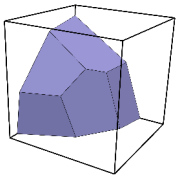
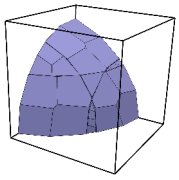
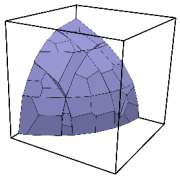
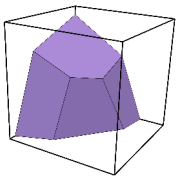
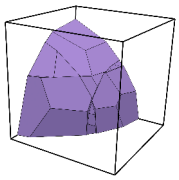
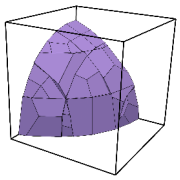
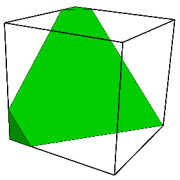
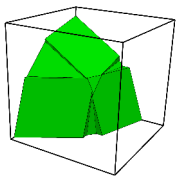
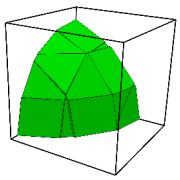
	$1 \times 1 \times 1$	$2 \times 2 \times 2$	$3 \times 3 \times 3$
MOF^3_{6hs}			
$\Sigma S(\Omega_r)$	6.96227×10^{-5}	3.6633×10^{-5}	2.7864×10^{-5}
MOF^3_{5hs}			
$\Sigma S(\Omega_r)$	9.5327×10^{-5}	2.8328×10^{-5}	1.8954×10^{-5}
MOF^3_{4hs}			
$\Sigma S(\Omega_r)$	1.2266×10^{-4}	4.3528×10^{-5}	2.2858×10^{-5}
MOF^3_{1hs}			
$\Sigma S(\Omega_r)$	4.8481×10^{-4}	1.0026×10^{-4}	4.8324×10^{-5}

Figure 7.18: Error comparison for a single cell across varying levels of refinement. We can see that additional levels of refinement are needed to capture important geometric features accurately, particularly in areas of high or variable curvature.

approximation that, visually at least, appears to be a poor approximation of the natural crease in the ground-truth shape. We investigate this particular cell in Figure 7.20, where we see that the complex geometric features of the material fragment make both the convex and convex-complement

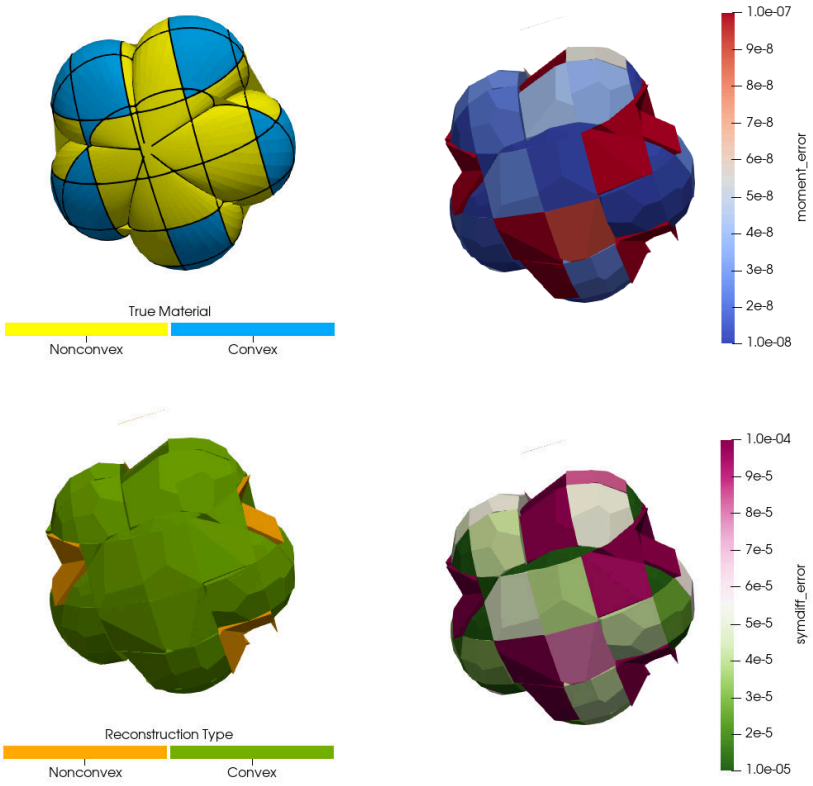
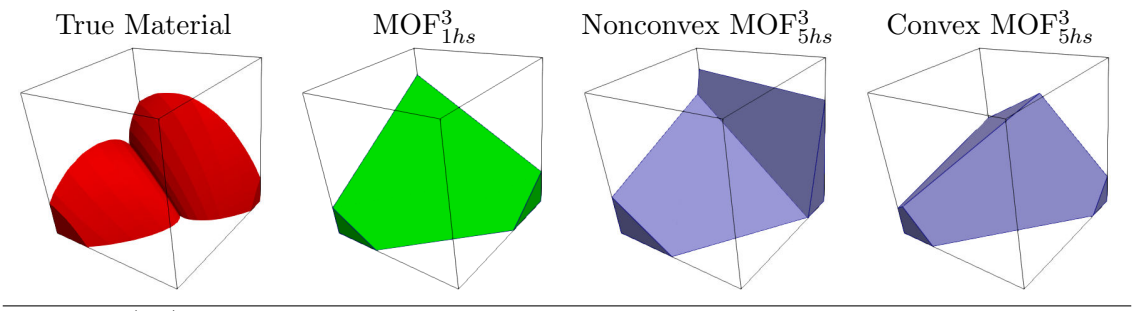


Figure 7.19: Reconstruction of material that is neither convex nor has a convex complement. Although the proposed technique can effectively approximate convex material, it is difficult to use such a reconstruction strategy to represent material which is neither convex nor the complement of a convex shape, as shown by the increased error for such cells.

reconstructions largely undesirable. Nevertheless, we see that the convex reconstruction (which has functionally combined the two components of ground-truth material) provides a better approximation along each of the two measured error metrics. We also see in this single cell example that both trial materials generated by MOF^3_{5hs} outperform the PLIC reconstruction, as such a technique can (at best) only accurately represent convex material in a convex cell. In either case, we can see throughout the provided examples that the multi-plane method dramatically outperforms the PLIC method numerically, despite the result appearing visually unintuitive.



$E(\Omega_r)$	0.069195	0.0248674	0.0223386
$S(\Omega_r)$	0.125328	0.11804	0.0915591

Figure 7.20: Reconstruction of a single POM that is neither convex nor has a convex complement. Although the selection of a nonconvex material appears unintuitive for this type of shape, it nevertheless provides a considerably lower error both in terms of the moment error and the symmetric difference error.

7.7.2 Geometric Artifacts

In each of the proposed MOF methods, we occasionally observe that one or more planes used in the reconstruction only exclude a very small volume from the convex intersections of half-spaces that represents the approximated material. When the material itself is convex, these artifacts manifests through "gaps" or "holes" between adjacent cells that are otherwise filled in the ground-truth shape. The same type of artifact is more prominent visually, however, in the case where the material is non-convex, such as that emphasized in cyan in Figure 7.16, where thin "slices" of material appear catastrophic for the quality of the interface.

The primary source of these artifacts is mathematical. The geometric moments used in the objective function vary continuously with our parameterization of the reconstructed interface with no direct influence from cell boundaries. These "gaps" or "slices" in the reconstruction have near-zero volume, and so this means that during optimization, there is no little to no distinction between returned interfaces that have or lack such artifacts, as they functionally do not contribute to the error.

From the perspective of the reconstruction problem, we consider these artifacts to be a simple visual error whose visually striking appearance can be attributed almost entirely to the particular viewing angle at which the shape is observed.

Similarly from a physical perspective, they can be expected to be functionally inert with respect to Lagrangian remapping as well, as the fact that they have near-zero volume means their individual moments do not meaningfully contribute to the intersection with a backtraced cell. However, we acknowledge that there are circumstances in which material interactions between the *surface* of such spurious components can be impactful for physics applications.

We consider the more rigorous treatment of these surface interactions to be separate from the reconstruction problem, but we nevertheless suggest the approach of straight-forwardly removing from the reconstruction any plane that only clips a small volume from the cell. By weighting this cutoff volume by the reference material volume, we also ensure that cells with *only* small volume components are not completely removed. While by definition this procedure only removes material with otherwise insignificant volume, the total volume within the cell must still be tightly controlled to preserve physically in numerical simulation. In such cases where the volume has diverged from the reference, we can re-enforce the volume constraint on the remaining clipping planes through the same root-finding problem employed during optimization at a cost that is less than that of a single optimization step (See Section 7.4.2). We demonstrate the results of this strategy on each of the proposed family of methods on an $8 \times 8 \times 8$ grid of cells for the shape in Figure 7.16. In Figure 7.21 we see that even an extremely conservative tolerance for the cut volume works well to remove these artifacts.

7.7.3 Shapes with Identical Moments

Although the entire set of geometric moments is sufficient to uniquely identify any shape, numerical restrictions require that a method such as ours only consider a finite subset of available moment data during optimization. This leads to cases in which multiple geometrically distinct shapes have identical moments up to a certain order, typically when the shape itself posses symmetry of some kind. A possible solution would be to simply use additional high-order moments in the construction of the interface to distinguish such shapes. Indeed, it is for this reason that we only propose MOF methods that use, at minimum, third-order moments, as we observe empirically that

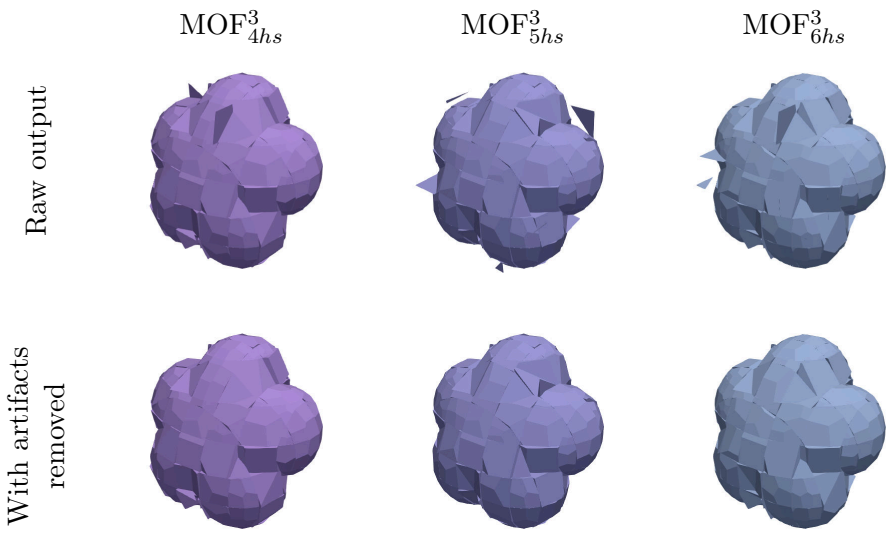


Figure 7.21: Example of visual artifact removal. Because the optimization procedure is indifferent to the number of components that compose each material in a cell, there are often superficial artifacts introduced by bounding planes that define regions of near-zero volume. This is especially noticeable in the case of convex-complement reconstructions, as such artifacts have the appearance of thin sheets or lines. Nevertheless, we consider these extraneous features to be largely irrelevant to downstream physics applications, and suggest a simple strategy for removing them.

there exist too many possible symmetries for geometric objects in 3D for second-order moments to be sufficient.

However, we note that even using third-order moments, there are many shapes that remain indistinguishable. For example, this can be observed in any regular cube inscribed in the same sphere (See Figure 7.22), all of which have identical moments up to and including third-order. Indeed, any MOF$^3_{6hs}$ method attempted on these shapes will terminate immediately, as the initial inscribed cube matches the *available* moments exactly.

At the same time, we have observed that considering *more* than third-order moments also leads to undesirable results, likely because fourth-order moments individually encode less important information than quantities derived from lower-order moments like the centroid. This is particularly problematic when the proposed optimization procedure weights all moments equally. For example, a reconstruction that more closely matches the 15 fourth-order moments than the 10 third-order moments will likely be less accurate to the underlying geometry.

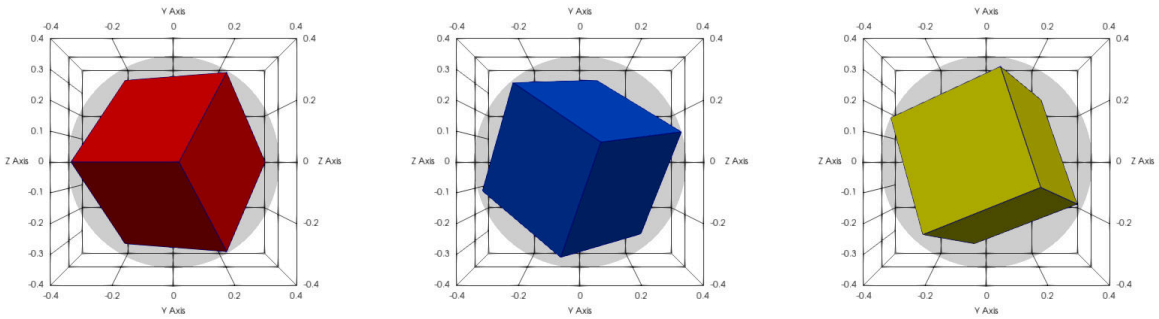


Figure 7.22: Example of distinct shapes with some identical moments. These shapes have identical moments up to third order, and can therefore not be distinguished by any of the proposed methods.

Nevertheless, we observe throughout this work that cases where moments up to third-order are incapable of distinguishing shapes are largely adversarial, and the reconstructions generated by the provided MOF^3 methods are suitably capable of expressing realistic geometric features.

7.7.4 Sensitivity to Noise in Reference Moment Data

In a practical application, the provided reference moments are subject to numerical error. For example, one possible source of error is the physics simulation scheme under which the material evolves over time, from which moment data is calculated. This causes the material itself to erroneously drift, which impacts the accuracy of the interface reconstruction relative to the unknown ground truth.

Another potentially more manageable source is the calculation of moment data itself from the polyhedral approximation after each step of the simulation. Here too the accuracy of the interface reconstruction method is affected, but now instead relative to data from a previous state of the simulation. While this error cannot be accounted for by interface reconstruction techniques, it is important that the non-linear optimization scheme used by the proposed family of methods is not unnecessarily sensitive to slight numerical issues with reference moment data.

To demonstrate this, we consider a sphere placed on a $3 \times 3 \times 3$ grid, a shape for which the exact shape cannot be recovered exactly. The sphere itself is centered at $(0.1, -0.1, 0.1)$ and has a

radius of 0.7 to avoid unfairly exploiting the symmetry of the shape in the reconstruction.

After computing ground-truth moment data for this shape, we add an increasing amount of normally distributed noise to each raw moment to simulate this second kind of numerical error. We then perform a reconstruction with the proposed family of methods, as well as the simple MOF^3_{1hs} PLIC method for further comparison. Specifically, we add a sample n_i from the normal distribution $\mathcal{N}(0, 10^{-L})$ to each moment for varying levels of noise L, in effect perturbing the moment data in the L^{th} decimal place. For context, the ground-truth moments vary amongst themselves on the order of $10^{-3} \sim 10^{-5}$. We compile these results for various levels of noise in Figure 7.23. The first row of noisy results shows a fairly realistic case, where a noise level $L = 8$ corresponds roughly to the rounding that would be observed if double precision values were converted to single precision. This is essentially a perturbation in the 6^{th} significant figure. As shown in the figure, this has functionally no impact on the accuracy of the method.

By $L = 5$, the inaccuracy borders on unrealistic, as we have now perturbed most pieces of data in at least their 3^{rd} significant figure. By this point the quality of the reconstruction has suffered significantly, particularly with the increased presence of visible geometric artifacts stemming from the use of non-conex reconstructions, yet we nevertheless can recognize the original sphere shape. By the 3^{rd} level of noise, all values have zero digits of accuracy after perturbation, and it is only at this point does the shape become completely unrecognizable and unusable. Altogether, these results demonstrate that the non-linear optimization scheme is fairly robust to imprecision in moment data, and more generally, that moment information is a useful method of encoding geometric features of polyhedra.

7.8 Conclusions and Future Work

In this work we have presented a novel approach to 3D interface reconstruction, in which we approximate material as the convex intersection (or the complement thereof) of 4, 5, and 6 half-spaces. The parameters of each half-space are selected through the minimization of a non-linear optimization problem over zeroth-order through third-order reference moments. In many ways,

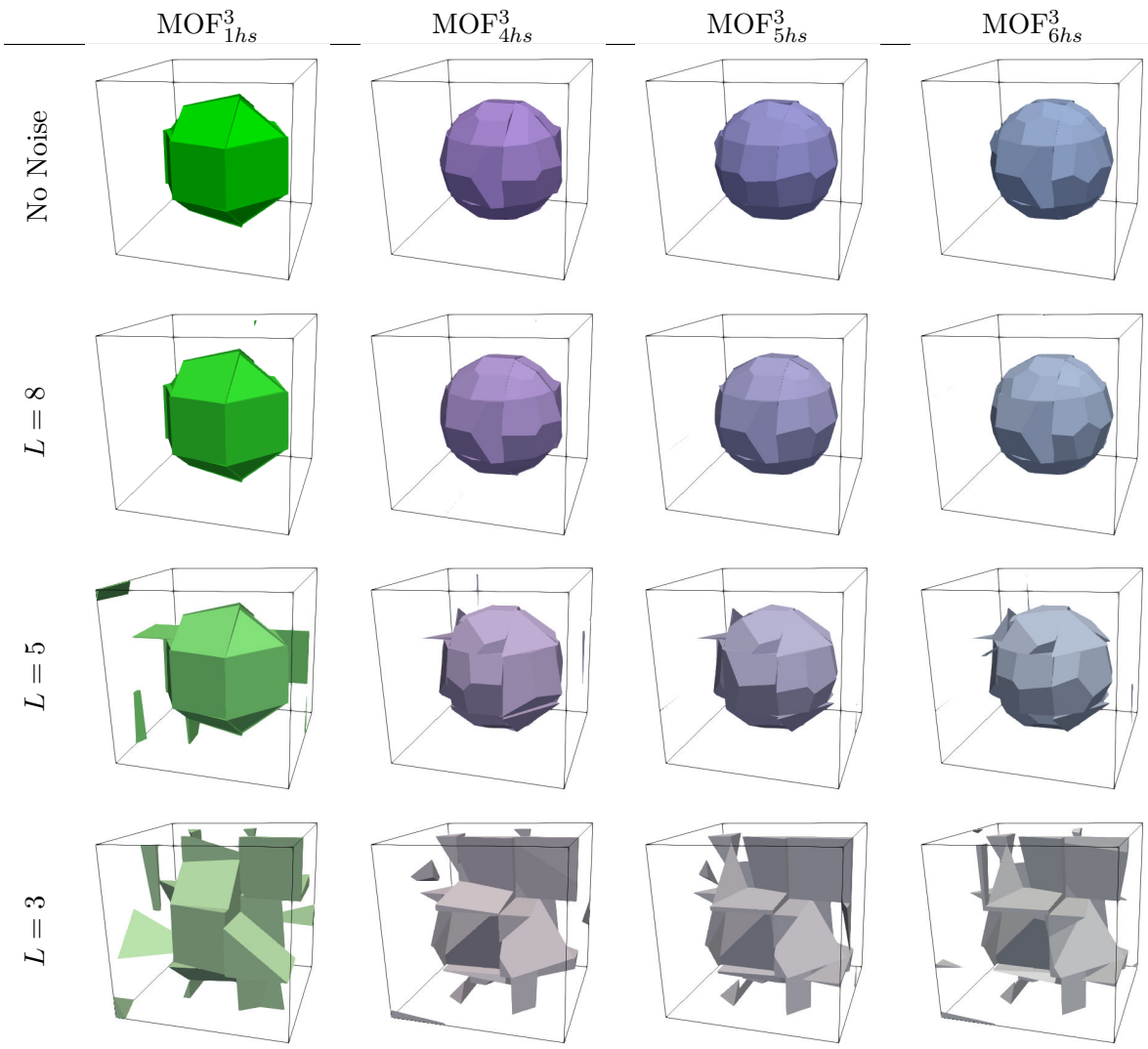


Figure 7.23: Example of the effect of noise in moment data on interface reconstruction. For noise level L, we add to each piece of moment data a random sample from the normal distribution $\mathcal{N}(0, 10^{-L})$. While the presence of noise in reference moment data makes it impossible to reconstruct the shape accurately with respect to the ground truth, our non-linear optimization scheme is nevertheless stable enough to withstand small perturbations in moment data before becoming unusable.

this work represents the natural progression of the existing body of literature in 2D multi-plane interface reconstruction, where we have combined the geometric expressiveness derived from using multiple independently parameterized planes (as in MOF$^2_{2hp}$) with the computational efficiency derived from using a single initial condition for the non-linear optimization (as in MOF$^3_{PIE}$). At the same time, our approach is aided by a custom normalization procedure that uses data from a

reference ellipsoid to transform the space over which optimization is performed, thereby improving robustness and computational efficiency of the method. Through these techniques, we are able to create material approximations that far outpace PLIC methods on the same level of resolution, in particular being able to more accurately resolve areas of high curvature or sharp features.

The development of this family of techniques leaves much room for further improvements, both computationally and algorithmically. For example, the largest barrier to reaching a true global minimum as opposed to a somewhat acceptable local minimum during optimization is the presence of "inactive planes," in which one of the planes reaches a plateau of the objective function, for which the local gradient is zero. In such cases, the plane becomes fixed in this position for the remainder of the optimization procedure. While the current approach of "clamping" planes to be tangent to the cell alleviates this issue, it is an *ad hoc* solution that could be improved in future work. Furthermore, while the generic Levenberg-Marquardt algorithm used for numerical optimization is effective at solving this problem, it is possible that more specified procedures could achieve an improved optimal solution at a faster rate, particularly those that utilize automatic differentiation.

Finally, there exist many existing techniques in the MOF literature which could be more-or-less immediately applied to the proposed multi-plane method. As described in Section 7.2.2, defining the interface with half-spaces makes the proposed method highly compatible with contemporary advection strategies, and in principle generalizes well to multi-material scenarios.

Additionally, one could improve computational performance across a mesh through a method that adaptively selects the number of planes to use in the reconstruction, as areas of low curvature are often sufficiently captured by a single plane, while the advantages between MOF^3_{4hs}, MOF^3_{5hs}, and MOF^3_{6hs} increase with increasing curvature.

7.9 Acknowledgement

The work of this chapter was done under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory under Contract No. 89233218CNA000001. The authors gratefully acknowledges the support of the US Depart-

ment of Energy National Nuclear Security Administration Advanced Simulation and Computing Program. LA-UR-24-30258.

Bibliography

[1] Anirudh Acharya, Theodore Kypraios, and Mădălin Guţă. A comparative study of estimation methods in quantum tomography. Journal of Physics A: Mathematical and Theoretical, 52(23):234001, May 2019.

[2] Hyung Taek Ahn and Mikhail Shashkov. Multi-material interface reconstruction on generalized polyhedral meshes. Journal of Computational Physics, 226(2):2096–2132, 2007.

[3] Rehman Ali, Carl D. Herickhoff, Dongwoon Hyun, Jeremy J. Dahl, and Nick Bottenus. Extending retrospective Encoding for Robust Recovery of the Multistatic Data Set. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 67(5):943–956, 2020.

[4] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, Tz. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini. MFEM: A modular finite element methods library. Computers & Mathematics with Applications, 81:42–74, jan 2021.

[5] P. Asente, M. Schuster, and T. Pettit. Dynamic planar map illustration. ACM Transactions on Graphics, 26(3):30:1–10, July 2007.

[6] Anirudh Asuri Mukundan, Thibaut Ménard, Jorge César Brändle de Motta, and Alain Berlemont. A 3d moment of fluid method for simulating complex turbulent multiphase flows. Computers & Fluids, 198:104364, 2020.

[7] E. Aulisa, S. Manservisi, R. Scardovelli, and S. Zaleski. Interface reconstruction with least-squares fit and split advection in three-dimensional cartesian geometry. Journal of Computational Physics, 225(2):2301–2319, 2007.

[8] A. Balu, M. R. Rajanna, J. Khristy, F. Xu, A. Krishnamurthy, and M.-C. Hsu. Direct immersogeometric fluid flow and heat transfer analysis of objects represented by point clouds. Computer Methods in Applied Mechanics and Engineering, 404:115742, February 2023.

[9] K. Banaszek, G. M. D'Ariano, M. G. A. Paris, and M. F. Sacchi. Maximum-likelihood estimation of the density matrix. Phys. Rev. A, 61:010304, Dec 1999.

[10] G. Barill, N. Dickson, R. Schmidt, D. I. W. Levin, and A. Jacobson. Fast winding numbers for soups and clouds. ACM Transactions on Graphics, 37(4), July 2018.

[11] A. J. Barlow, P.-H. Maire, W. J. Rider, R. N. Rieben, and M. J. Shashkov. Arbitrary Lagrangian–Eulerian methods for modeling high-speed compressible multimaterial flows. Journal of Computational Physics, 322:603–665, 2016.

[12] F. Bassi and S. Rebay. High-order accurate discontinuous finite element solution of the 2d euler equations. Journal of Computational Physics, 138(2):251–285, 1997.

[13] T. Baumgratz, D. Gross, M. Cramer, and M. B. Plenio. Scalable reconstruction of density matrices. Phys. Rev. Lett., 111:020401, Jul 2013.

[14] T Baumgratz, A Nüßeler, M Cramer, and M B Plenio. A scalable maximum likelihood method for quantum state tomography. New Journal of Physics, 15(12):125004, dec 2013.

[15] Sayantan Bhadra, Varun A. Kelkar, Frank J. Brooks, and Mark A. Anastasio. On hallucinations in tomographic image reconstruction. IEEE Transactions on Medical Imaging, 40(11):3249–3260, 2021.

[16] S. Bischoff, D. Pavic, and L. Kobbelt. Automatic restoration of polygon models. ACM Transactions on Graphics, 24(4):1332–1352, October 2005.

[17] Robin Blume-Kohout. Hedged maximum likelihood quantum state estimation. Physical Review Letters, 105(20), November 2010.

[18] Robin Blume-Kohout. Optimal, reliable estimation of quantum states. New Journal of Physics, 12(4):043034, apr 2010.

[19] Nick Bottenus. Recovery of the complete data set from focused transmit beams. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 65(1):30–38, 2018.

[20] Nick Bottenus, Jacob Spainhour, and Stephen Becker. Comparison of spatial encodings for ultrasound imaging. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 70(1):52–63, 2023.

[21] J. C. Bowers, J. Leahey, and R. Wang. A ray tracing approach to diffusion curves. Computer Graphics Forum, 30(4):1345–1352, 2011.

[22] Andrew Cahaly, Fabien Evrard, and Olivier Desjardins. Plic-net: A machine learning approach for 3d interface reconstruction in volume of fluid methods. International Journal of Multiphase Flow, 178:104888, Aug 2024.

[23] A. Capps, R. Carson, B. Corbett, N. Elliott, J. Essman, B. Gunney, B. Han, C. Harrison, R. Hornung, M. Larsen, A. Moody, E. Pauli, R. Settgast, L. Taylor, K. Weiss, C. White, B. Whitlock, M. Yang, and G. Zagaris. Axom: CS infrastructure components for HPC applications, 2017–2024. `https://github.com/llnl/axom`.

[24] P. C. P. Carvalho and P. R. Cavalcanti. Ii.2 - point in polyhedron testing using spherical polygons. In A. W. Paeth, editor, Graphics Gems V, pages 42–49. Academic Press, Boston, 1995.

[25] Marco Cerezo, Alexander Poremba, Lukasz Cincio, and Patrick J. Coles. Variational quantum fidelity estimation. Quantum, 4:248, March 2020.

[26] Yinran Chen, Jing Liu, Xiongbiao Luo, and Jianwen Luo. ApodNet: Learning for High Frame Rate Synthetic Transmit Aperture Ultrasound Imaging. IEEE Transactions on Medical Imaging, 40(11):3190–3204, 2021.

[27] Yujie Chen, Junhua Gong, Dongliang Sun, Dongxu Han, Peng Wang, Bo Yu, and Wen-Quan Tao. A three-dimensional curve interface reconstruction algorithm for two-phase fluid flow. Journal of Computational Physics, 520:113489, 2025.

[28] RY Chiao, LJ Thomas, and SD Silverstein. Sparse array imaging with spatially-encoded transmits. 1997 IEEE Ultrasonics Symposium, pages 1679–1682, 1997.

[29] E. B. Chin and N. Sukumar. Scaled boundary cubature scheme for numerical integration over planar regions with affine and curved boundaries. Computer Methods in Applied Mechanics and Engineering, 380:113796, 2021.

[30] Robert Chiodi and Olivier Desjardins. General, robust, and efficient polyhedron intersection in the interface reconstruction library. Journal of Computational Physics, 449:110787, Oct 2021.

[31] Robert Chiodi and Mikhail Shashkov. A moment-of-fluid interface reconstruction using polygon inscribed in ellipse in 2d. Journal of Computational Physics, 528:113814, 2025.

[32] Jon Claerbout. Geophysical image estimation by example. LULU COM, 2014.

[33] J. E. Cobb. Tiling the sphere with rational Bézier patches. In TR UUCS-88-009, pages 1–14. University of Utah USA, 1988.

[34] R S C Cobbold. Foundations of Biomedical Ultrasound. Biomedical Engineering Series. Oxford University Press, 2007.

[35] P. Colella, D. Graves, T. Ligocki, D. Trebotich, and B. V. Straalen. Embedded boundary algorithms and software for partial differential equations. Journal of Physics: Conference Series, 125(1):012084, July 2008.

[36] Marcus Cramer, Martin B Plenio, Steven T Flammia, Rolando Somma, David Gross, Stephen D Bartlett, Olivier Landon-Cardinal, David Poulin, and Yi-Kai Liu. Efficient quantum state tomography. Nature communications, 1(1):149, 2010.

[37] Marcus P. da Silva, Olivier Landon-Cardinal, and David Poulin. Practical Characterization of Quantum Devices without Tomography. Phys. Rev. Lett., 107(21):210404, 2011.

[38] M. Dowell and P. Jarratt. A modified regula falsi method for computing the root of an equation. BIT, 11(2):168–174, June 1971.

[39] Vadim Dyadechko and Mikhail Shashkov. Moment-of-fluid interface reconstruction. Technical report, Los Alamos National Laboratory, Los Alamos, NM, Oct 2005. Technical Report LA-UR-05-7571.

[40] Vadim Dyadechko and Mikhail Shashkov. Reconstruction of multi-material interfaces from moment data. Journal of Computational Physics, 227(11):5361–5384, 2008.

[41] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. ACM Trans. Graph., 9(1):66–104, January 1990.

[42] A. Efremov, V. Havran, and H.-P. Seidel. Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces. In Proceedings of the 21st Spring Conference on Computer Graphics, SCCG '05, pages 127–135, New York, NY, USA, 2005. Association for Computing Machinery.

[43] A. Efremov, V. Havran, and H.-P. Seidel. Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces. In Proceedings of the 21st Spring Conference on Computer Graphics, SCCG '05, pages 127–135, New York, NY, USA, 2005. Association for Computing Machinery.

[44] Fabien Evrard, Robert Chiodi, Austin Han, Berend van Wachem, and Olivier Desjardins. First moments of a polyhedron clipped by a paraboloid. SIAM Journal on Scientific Computing, 45(5):A2250–A2274, 2023.

[45] Fabien Evrard, Robert Chiodi, Berend van Wachem, and Olivier Desjardins. Simulating interfacial flows: a farewell to planes. arXiv preprint arXiv:2401.15012, 2024.

[46] G. E. Farin. Curves and surfaces for CAGD: A practical guide. Morgan Kaufmann, 2001.

[47] N. Feng, M. Gillespie, and K. Crane. Perspectives on winding numbers. Technical report, Carnegie Mellon University, 2023.

[48] Nicole Feng and Keenan Crane. A heat method for generalized signed distance. ACM Trans. Graph., 43(4), July 2024.

[49] Jaromír Fiurášek and Zden ěk Hradil. Maximum-likelihood estimation of quantum processes. Phys. Rev. A, 63:020101, Jan 2001.

[50] Steven T. Flammia and Yi-Kai Liu. Direct fidelity estimation from few pauli measurements. Physical Review Letters, 106(23), jun 2011.

[51] Jan Flusser, Barbara Zitova, and Tomas Suk. Moments and Moment Invariants in Pattern Recognition. Wiley Publishing, 2009.

[52] J. E. Fromm, N. Wunsch, R. Xiang, H. Zhao, K. Maute, J. A. Evans, and D. Kamensky. Interpolation-based immersed finite element and isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, 405:115890, 2023.

[53] I. Ginzburg and G. Wittum. Two-phase flows on interface refined grids modeled with vof, staggered finite volumes, and spline interpolants. Journal of Computational Physics, 166(2):302–335, 2001.

[54] Sobhan Goudarzi and Hassan Rivaz. Deep reconstruction of high-quality ultrasound images from raw plane-wave data: A simulation and in vivo study. Ultrasonics, 125:106778, 2022.

[55] Andreas Griewank and Andrea Walther. Evaluating Derivatives. Society for Industrial and Applied Mathematics, second edition, 2008.

[56] David Gross, Yi-Kai Liu, Steven T. Flammia, Stephen Becker, and Jens Eisert. Quantum state tomography via compressed sensing. Phys. Rev. Lett., 105:150401, Oct 2010.

[57] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[58] D. Gunderman. High-Order Spatial Discretization and Numerical Integration Schemes for Curved Geometries. PhD thesis, Department of Applied Mathematics, University of Colorado Boulder, 2021.

[59] D. Gunderman, K. Weiss, and J. A. Evans. Spectral mesh-free quadrature for planar regions bounded by rational parametric curves. Computer-Aided Design, 130, 2020.

[60] D. Gunderman, K. Weiss, and J. A. Evans. High-accuracy mesh-free quadrature for trimmed parametric surfaces and volumes. Computer-Aided Design, 141:103093, 2021.

[61] D. Gunderman, K. Weiss, and J. A. Evans. High-accuracy mesh-free quadrature for trimmed parametric surfaces and volumes. Computer-Aided Design, 141:103093, December 2021.

[62] M Guţă, J Kahn, R Kueng, and J A Tropp. Fast state tomography with optimal error bounds. Journal of Physics A: Mathematical and Theoretical, 53(20):204001, apr 2020.

[63] C. Hafner, C. Schumacher, E. Knoop, T. Auzinger, B. Bickel, and M. Bächer. X-CAD: Optimizing CAD models with extended finite elements. ACM Trans. Graph., 38(6), November 2019.

[64] E. Haines. I.4. - point in polygon strategies. In P. S. Heckbert, editor, Graphics Gems, pages 24–46. Academic Press, 1994.

[65] E. Haines and T. Akenine-Möller, editors. Ray tracing gems. Apress, 2019. `http://raytracinggems.com`.

[66] Austin Han, Robert Chiodi, and Olivier Desjardins. Capturing thin structures in vof simulations with two-plane reconstruction, 2024.

[67] T Harrison, A Samplaleanu, and Roger Zemp. S-sequence spatially-encoded synthetic aperture ultrasound imaging. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 61(5):886–890, 2014.

[68] Philippe Hergibo, Timothy N. Phillips, and Zhihua Xie. A moment-of-fluid method for resolving filamentary structures using a symmetric multi-material approach. Journal of Computational Physics, 491:112401, 2023.

[69] C.W. Hirt, A. A. Amsden, and J. L. Cook. An Arbitrary Lagrangian-Eulerian computing method for all flow speeds. Journal of Computational Physics, 14(3):227–253, 1974.

[70] K. Hormann and A. Agathos. The point in polygon problem for arbitrary polygons. Computational Geometry, 20(3):131–144, 2001.

[71] Addison Howard, Eunbyung Park, and Wendy Kan. Imagenet object localization challenge, 2018.

[72] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema, and A. Krishnamurthy. Direct immersogeometric fluid flow analysis using B-rep CAD models. Computer Aided Geometric Design, 43:143–158, 2016. Geometric Modeling and Processing 2016.

[73] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering, 194(39):4135–4195, 2005.

[74] Dongwoon Hyun. A universal end-to-end description of pulse-echo ultrasound image reconstruction. In Stephen Aylward, J. Alison Noble, Yipeng Hu, Su-Lin Lee, Zachary Baum, and Zhe Min, editors, Simplifying Medical Ultrasound, pages 128–138, Cham, 2022. Springer International Publishing.

[75] Dongwoon Hyun, Leandra L. Brickson, Kevin T. Looby, and Jeremy J. Dahl. Beamforming and speckle reduction using neural networks. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 66(5):898–910, 2019.

[76] Dongwoon Hyun, Alycen Wiacek, Sobhan Goudarzi, Sven Rothlübbers, Amir Asif, Klaus Eickel, Yonina C. Eldar, Jiaqi Huang, Massimo Mischi, Hassan Rivaz, David Sinden, Ruud J. G. van Sloun, Hannah Strohm, and Muyinatu A. Lediju Bell. Deep learning for ultrasound image formation: Cubdl evaluation framework and open datasets. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 68(12):3466–3483, 2021.

[77] Jaroslav Řeháček, Zden ěk Hradil, E. Knill, and A. I. Lvovsky. Diluted maximum-likelihood algorithm for quantum tomography. Phys. Rev. A, 75:042108, Apr 2007.

[78] D. M. Ingram, D. M. Causon, and C. G. Mingham. Developments in Cartesian cut cell methods. Mathematics and Computers in Simulation, 61(3):561–572, 2003.

[79] A. Jacobson, L. Kavan, and O. Sorkine. Robust inside-outside segmentation using generalized winding numbers. ACM Trans. Graph., 32(4):1–12, 2013.

[80] A. Jacobson, D. Panozzo, C. Schüller, O. Diamanti, Q. Zhou, N. Pietroni, et al. libigl: A simple C++ geometry processing library. Google Scholar, 2013.

[81] Daniel F. V. James, Paul G. Kwiat, William J. Munro, and Andrew G. White. Measurement of qubits. Phys. Rev. A, 64:052312, Oct 2001.

[82] Matthew Jemison, Eva Loch, Mark Sussman, Mikhail Shashkov, Marco Arienti, Mitsuhiro Ohta, and Yaohong Wang. A coupled level set-moment of fluid method for incompressible two-phase flows. Journal of scientific computing, 54(2-3):38, 2013.

[83] J.A. Jensen. Simulation of advanced ultrasound systems using field ii. In 2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821), pages 636–639 Vol. 1, 2004.

[84] Jørgen Arendt Jensen, Svetoslav Ivanov Nikolov, Kim Løkke Gammelmark, and Morten Høgholm Pedersen. Synthetic aperture ultrasound imaging. Ultrasonics, 44:e5–e15, 2006.

[85] Mok Kun Jeong and Sung-Jae Kwon. A new method for assessing the performance of signal processing filters in suppressing the side lobe level. Ultrasonography, 40:289 – 300, 2020.

[86] Anatoli B Juditsky and Arkadi S Nemirovski. Nonparametric estimation by convex programming. The Annals of Statistics, 37(5A):2278 – 2300, 2009.

[87] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. Computer Methods in Applied Mechanics and Engineering, 284:1005–1053, 2015. Isogeometric Analysis Special Issue.

[88] M. J. Kilgard. Polar stroking: New theory and methods for stroking paths. ACM Transactions on Graphics, 39(4):145:1–15, August 2020.

[89] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[90] L. Klinteberg and A. H. Barnett. Accurate quadrature of nearly singular line integrals in two and three dimensions by singularity swapping. BIT Numerical Mathematics, 61:1–36, 07 2020.

[91] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo. Abc: A big CAD model dataset for geometric deep learning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Commun. ACM, 60(6):84–90, may 2017.

[93] Johannes Kromer and Dieter Bothe. Face-based volume-of-fluid interface positioning in arbitrary polyhedra. Journal of Computational Physics, 449:110776, 2022.

[94] Murali K. Kurmapu, V.V. Tiunova, E.S. Tiunov, Martin Ringbauer, Christine Maier, Rainer Blatt, Thomas Monz, Aleksey K. Fedorov, and A.I. Lvovsky. Reconstructing complex states of a 20-qubit quantum simulator. PRX Quantum, 4:040345, Dec 2023.

[95] Magnus Dalen Kvalevåg, Anders Emil Vrålstad, Ole Marius Hoel Rindal, Tore Grüner Bjåstad, Bastien Denarie, Kjell Kristoffersen, Svein-Erik Måsøy, and Lasse Løvstakken. vbeam: a fast and differentiable beamformer for optimizing ultrasound imaging. In 2023 IEEE International Ultrasonics Symposium (IUS), pages 1–4, 2023.

[96] Hannah Lange, Matjaž Kebrič, Maximilian Buser, Ulrich Schollwöck, Fabian Grusdt, and Annabelle Bohrdt. Adaptive quantum state tomography with active learning. Quantum, 7:1129, October 2023.

[97] Ben P Lanyon, Christine Maier, Milan Holzäpfel, Tillmann Baumgratz, Cornelius Hempel, Petar Jurcevic, Ish Dhand, AS Buyskikh, Andrew J Daley, Marcus Cramer, et al. Efficient tomography of a quantum many-body system. Nature Physics, 13(12):1158–1162, 2017.

[98] Weichao Liang, Francesco Ticozzi, and Giuseppe Vallone. Optimizing measurements sequences for quantum state verification. Quantum Information Processing, 22(11), November 2023.

[99] Alexander Lidiak, Casey Jameson, Zhen Qin, Gongguo Tang, Michael B Wakin, Zhihui Zhu, and Zhexuan Gong. Quantum state tomography with tensor train cross approximation. arXiv preprint arXiv:2207.06397, 2022.

[100] Shengfeng Liu, Yi Wang, Xin Yang, Baiying Lei, Li Liu, Shawn Xiang Li, Dong Ni, and Tianfu Wang. Deep learning in medical ultrasound analysis: A review. Engineering, 5(2):261–275, 2019.

[101] Shengping Liu, Heng Yong, Shaodong Guo, Yiqing Shen, and Guoxi Ni. An improved continuity-preserving interface reconstruction method for multi-material flow. Computers & Fluids, 224:104960, 2021.

[102] J. López, J. Hernández, P. Gómez, and F. Faura. An improved plic-vof method for tracking thin fluid structures in incompressible two-phase flows. Journal of Computational Physics, 208(1):51–74, 2005.

[103] Y. L. Ma and W. T. Hewitt. Point inversion and projection for NURBS curve and surface: Control polygon approach. Computer Aided Geometric Design, 20(2):79–99, 2003.

[104] Ram Kumar Maity, T. Sundararajan, and K. Velusamy. An accurate interface reconstruction method using piecewise circular arcs. International Journal for Numerical Methods in Fluids, 93(1):93–126, 2021.

[105] E. Marchandise, C. Piret, and J.-F. Remacle. CAD and mesh repair with Radial Basis Functions. Journal of Computational Physics, 231(5):2376–2387, 2012.

[106] Tomislav Marić, Douglas B. Kothe, and Dieter Bothe. Unstructured un-split geometrical volume-of-fluid methods – a review. Journal of Computational Physics, 420:109695, 2020.

[107] Donald Marquardt. An algorithm for least square estimation of non-linear parameters. SIAM Journal on Applied Mathematics, 11:431–441, Jun 1963.

[108] Z. Marschner, P. Zhang, D. Palmer, and J. Solomon. Sum-of-squares geometry processing. ACM Transactions on Graphics, 40(6), December 2021.

[109] Cedric Martens and Mikhail Bessmeltsev. One-shot method for computing generalized winding numbers, 2024.

[110] W. Martin, E. Cohen, R. Fish, and P. Shirley. Practical ray tracing of trimmed NURBS surface. Journal of Graphics Tools, 5, 09 2000.

[111] B. Marussig and T. Hughes. A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects. Archives of Computational Methods in Engineering, 25:1–69, 06 2017.

[112] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. Efficient elasticity for character skinning with contact and collisions. ACM Trans. Graph., 30(4), July 2011.

[113] A. A. Mezentsev and T. Woehler. Methods and algorithms of automated CAD repair for incremental surface meshing. In IMR, pages 299–309. Citeseer, 1999.

[114] Thomas Milcent and Antoine Lemoine. Moment-of-fluid analytic reconstruction on 3d rectangular hexahedrons. Journal of Computational Physics, 409:109346, 2020.

[115] Massimo Mischi, Muyinatu A. Lediju Bell, Ruud J. G. van Sloun, and Yonina C. Eldar. Deep learning in medical ultrasound—from image formation to image analysis. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 67(12):2477–2480, 2020.

[116] Mohammadreza Mohammadi and Agata M. Brańczyk. Optimization of quantum state tomography in the presence of experimental constraints. Phys. Rev. A, 89:012113, Jan 2014.

[117] Stewart Mosso, Christopher Garasi, and Richard Drake. A smoothed two- and three-dimensional interface reconstruction method. Comput. Vis. Sci., 12(7):365–381, Sep 2009.

[118] D. Nehab and H. Hoppe. Random-access rendering of general vector graphics. ACM Transactions on Graphics, 27(5), December 2008.

[119] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 2010.

[120] T. Nishita, T. W. Sederberg, and M. Kakimoto. Ray tracing trimmed rational surface patches. SIGGRAPH Comput. Graph., 24(4):337–345, sep 1990.

[121] C. R. Noble, A. T. Anderson, N. R. Barton, J. A. Bramwell, A. Capps, M. H. Chang, J. J. Chou, D. M. Dawson, E. R. Diana, T. A. Dunn, D. R. Faux, A. C. Fisher, P. T. Greene, I. Heinz, Y. Kanarska, S. A. Khairallah, B. T. Liu, J. D. Margraf, A. L. Nichols, R. N. Nourgaliev, M. A. Puso, J. F. Reus, P. B. Robinson, A. I. Shestakov, J. M. Solberg, D. Taller, P. H. Tsuji, C. A. White, and J. L. White. ALE3D: An Arbitrary Lagrangian-Eulerian multi-physics code. Technical report, Lawrence Livermore National Laboratory, 5 2017.

[122] Takumi Noda, Naoki Tomii, Keiichi Nakagawa, Takashi Azuma, and Ichiro Sakuma. Shape estimation algorithm for ultrasound imaging by flexible array transducer. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, PP:1–1, 06 2020.

[123] F. S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. IEEE Transactions on Visualization and Computer Graphics, 9(2):191–205, 2003.

[124] Open Cascade SAS. Open cascade technology. `https://dev.opencascade.org/`, 2011. Version 7.8.

[125] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin. Diffusion curves: A vector representation for smooth-shaded images. ACM Transactions on Graphics, 27(3):92:1–8, August 2008.

[126] Sam Pallister, Noah Linden, and Ashley Montanaro. Optimal verification of entangled states with local measurements. Phys. Rev. Lett., 120:170502, Apr 2018.

[127] M. A. Park, R. Haimes, N. J. Wyman, P. A. Baker, and A. Loseille. Boundary representation tolerance impacts on mesh generation and adaptation. In AIAA AVIATION 2021 FORUM, 2021.

[128] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[129] N. Patrikalakis and T. Maekawa. Shape Interrogation for Computer Aided Design and Manufacturing. 01 2010.

[130] Vincent Perrot, Maxime Polichetti, François Varray, and Damien Garcia. So you think you can DAS? a viewpoint on delay-and-sum beamforming. Ultrasonics, 111:106309, mar 2021.

[131] C. S. Peskin. The immersed boundary method. Acta Numerica, 11:479–517, 2002.

[132] L. Piegl and W. Tiller. The NURBS book. Springer Science & Business Media, 1996.

[133] L. A. Piegl and A. M. Richard. Tessellating trimmed NURBS surfaces. Computer-Aided Design, 27(1):16–26, 1995.

[134] James Edward Pilliod and Elbridge Gerry Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. Journal of Computational Physics, 199(2):465–502, 2004.

[135] Glenn Robert Price. A piecewise parabolic volume tracking method for the numerical simulation of interfacial flows. University of Calgary Calgary, AB, 2000.

[136] Zhen Qin, Casey Jameson, Zhexuan Gong, Michael B. Wakin, and Zhihui Zhu. Optimal allocation of pauli measurements for low-rank quantum state tomography, 2024.

[137] Zhen Qin, Casey Jameson, Zhexuan Gong, Michael B. Wakin, and Zhihui Zhu. Quantum state tomography for matrix product density operators. IEEE Transactions on Information Theory, 70(7):5030–5056, July 2024.

[138] Karthik Ranganathan and William F. Walker. Cystic resolution: A performance metric for ultrasound imaging systems. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 54(4):782–792, 2007.

[139] Ronald A. Remmerswaal and Arthur E.P. Veldman. Parabolic interface reconstruction for 2d volume of fluid methods. Journal of Computational Physics, 469:111473, 2022.

[140] A. Reshetov. Cool patches: A geometric approach to ray/bilinear patch intersections, pages 95–109. Apress, Berkeley, CA, 2019.

[141] William J. Rider and Douglas B. Kothe. Reconstructing volume tracking. Journal of Computational Physics, 141(2):112–152, 1998.

[142] Alfonso Rodriguez-Molares, Ole Marius, Hoel Rindal, and D Jan. The Generalized Contrast-to-Noise ratio : a formal definition for lesion detectability. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, pages 1–12, 2019.

[143] Anet Sanchez, Isaac Martinez, Jacob Spainhour, and Nick Bottenus. An optically tracked platform for swept synthetic aperture ultrasound imaging. In 2024 IEEE Conference on Computational Imaging Using Synthetic Apertures (CISA), pages 01–05, 2024.

[144] R. Sawhney and K. Crane. Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. ACM Transactions on Graphics, 39(4), 2020.

[145] Ruben Scardovelli and Stephane Zaleski. Interface reconstruction with least-square fit and split eulerian–lagrangian advection. International Journal for Numerical Methods in Fluids, 41(3):251–274, 2003.

[146] A. Schollmeyer and B. Froehlich. Efficient and anti-aliased trimming for rendering large NURBS models. IEEE Transactions on Visualization and Computer Graphics, 25(3):1489–1498, 2019.

[147] A. Schollmeyer and B. Fröhlich. Direct trimming of NURBS surfaces on the GPU. ACM Transactions on Graphics, 28(3), July 2009.

[148] T. W. Sederberg, G. T. Finnigan, X. Li, H. Lin, and H. Ipson. Watertight trimmed NURBS. ACM Transactions on Graphics, 27(3):1–8, August 2008.

[149] T. W. Sederberg, G. T. Finnigan, X. Li, H. Lin, and H. Ipson. Watertight trimmed NURBS. In ACM SIGGRAPH 2008 Papers, SIGGRAPH '08, New York, NY, USA, 2008. Association for Computing Machinery.

[150] T. W. Sederberg and T. Nishita. Curve intersection using Bézier clipping. Computer-Aided Design, 22(9):538–549, 1990.

[151] S. Sellán and A. Jacobson. Stochastic poisson surface reconstruction. ACM Transactions on Graphics, 41(6), November 2022.

[152] Akshay Seshadri, Martin Ringbauer, Rainer Blatt, Thomas Monz, and Stephen Becker. Versatile fidelity estimation with confidence, 2021.

[153] Akshay Seshadri, Martin Ringbauer, Thomas Monz, and Stephen Becker. Theory of versatile fidelity estimation with confidence, 2021.

[154] R. Sevilla, S. Fernández-Méndez, and A. Huerta. NURBS-enhanced finite element method for Euler equations. International Journal for Numerical Methods in Fluids, 57(9):1051–1069, 2008.

[155] R. Sevilla, S. Fernández-Méndez, and A. Huerta. 3d NURBS-enhanced finite element method (NEFEM). International Journal for Numerical Methods in Engineering, 88(2):103–125, 2011.

[156] R. Sevilla and A. Huerta. HDG-NEFEM with degree adaptivity for Stokes flows. Journal of Scientific Computing, 77(3):1953–1980, February 2018.

[157] Mikhail Shashkov. An adaptive moments-based interface reconstruction using intersection of the cell with one half-plane, two half-planes and a circle. Journal of Computational Physics, 494:112504, 2023.

[158] Mikhail Shashkov and Eugene Kikinzon. Moments-based interface reconstruction, remap and advection. Journal of Computational Physics, 479:111998, 2023.

[159] J. Shen, L. Busé, P. Alliez, and N. Dodgson. A line/trimmed NURBS surface intersection algorithm using matrix representations. Computer Aided Geometric Design, 48:1–16, 2016.

[160] M. Shimrat. Algorithm 112: Position of point relative to polygon. Commun. ACM, 5(8):434, August 1962.

[161] B. M. Smith, T. J. Tautges, and P. P. H. Wilson. Sealing faceted surfaces to achieve watertight CAD models. In S. Shontz, editor, Proceedings of the 19th International Meshing Roundtable, pages 177–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[162] A. Sommariva and M. Vianello. inRS: Implementing the indicator function of NURBS-shaped planar domains. Applied Mathematics Letters, 130:108026, 2022.

[163] J. Spainhour, D. Gunderman, and K. Weiss. Robust containment queries over collections of rational parametric curves via generalized winding numbers. ACM Transactions on Graphics, 43(4), July 2024.

[164] Jacob Spainhour. Quadratic moment-of-fluid interface reconstruction. Florida State University Undergraduate Honors Thesis, Apr 2020.

[165] Jacob Spainhour, Stephen Becker, and Nick Bottenus. A strategy for synthetic aperture sequence design using numerical optimization. In 2022 IEEE International Ultrasonics Symposium (IUS), pages 1–4, 2022.

[166] Jacob Spainhour and Mikhail Shashkov. Multi-plane moment-of-fluid interface reconstruction in 3d, 09 2024.

[167] Jacob Spainhour, Korben Smart, Stephen Becker, and Nick Bottenus. Optimization of array encoding for ultrasound imaging. Physics in Medicine & Biology, 69(12):125024, jun 2024.

[168] Jacob Spainhour, Korben Smart, Stephen Becker, and Nick Bottenus. Source code and data repository for "optimization of array encoding for ultrasound imaging", 2024.

[169] Jacob Spainhour and Kenneth Weiss. Robust containment queries over collections of trimmed nurbs surfaces via generalized winding numbers, 2025.

[170] Severin Strobl, Arno Formella, and Thorsten Pöschel. Exact calculation of the overlap volume of spheres and mesh elements. Journal of Computational Physics, 311:158–172, 2016.

[171] D.L. Sun and W.Q. Tao. A coupled volume-of-fluid and level set (voset) method for computing incompressible two-phase flows. International Journal of Heat and Mass Transfer, 53(4):645–655, 2010.

[172] T. Sun, P. Thamjaroenporn, and C. Zheng. Fast multipole representation of diffusion curves and points. ACM Transactions on Graphics, 33(4), July 2014.

[173] D. C. Thomas, M. A. Scott, J. A. Evans, K. Tew, and E. J. Evans. Bézier projection: A unified approach for local projection and quadrature-free refinement and coarsening of NURBS and T-splines with particular application to isogeometric design and analysis. Computer Methods in Applied Mechanics and Engineering, 284:55–105, 2015. Isogeometric Analysis Special Issue.

[174] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. Numerical grid generation: Foundations and applications. Elsevier North-Holland, Inc., 1985.

[175] P. Trettner, J. Nehring-Wirxel, and L. Kobbelt. EMBER: Exact mesh booleans via efficient & robust local arrangements. ACM Trans. Graph., 41(4), July 2022.

[176] P. G. Tucker and Z. Pan. A Cartesian cut cell method for incompressible viscous flow. Applied Mathematical Modelling, 24(8):591–606, 2000.

[177] Ruud J. G. van Sloun, Regev Cohen, and Yonina C. Eldar. Deep learning in ultrasound imaging. Proceedings of the IEEE, 108(1):11–29, 2020.

[178] C. Wang, F. Xu, M.-C. Hsu, and A. Krishnamurthy. Rapid B-rep model preprocessing for immersogeometric analysis using analytic surfaces. Computer Aided Geometric Design, 52-53:190–204, 2017. Geometric Modeling and Processing 2017.

[179] Qisheng Wang, Zhicheng Zhang, Kean Chen, Ji Guan, Wang Fang, Junyi Liu, and Mingsheng Ying. Quantum algorithm for fidelity estimation. IEEE Transactions on Information Theory, 69(1):273–282, 2023.

[180] K. Weiss, G. Zagaris, R. Rieben, and A. Cook. Spatially accelerated shape embedding in multimaterial simulations. In S. Canann, editor, Proceedings 25^{th} International Meshing Roundtable, IMR '16, Washington, D.C., September 27–30 2016.

[181] Hans Wolters and Hewlett Laboratories. Extensions: Extrapolation methods for cad. 04 2000.

[182] C. Wyman and A. Marrs. Introduction to DirectX raytracing, pages 21–47. Apress, Berkeley, CA, 2019.

[183] R. Xiong, Y. Lu, C. Chen, J. Zhu, Y. Zeng, and L. Liu. ETER: Elastic tessellation for real-time pixel-accurate rendering of large-scale NURBS models. ACM Transactions on Graphics, 42(4), July 2023.

[184] Qi You, Zhijie Dong, Matthew R. Lowerison, and Pengfei Song. Pixel-oriented Adaptive Apodization for Planewave Imaging Based on Recovery of the Complete Data Set. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, pages 1–1, 2021.

[185] S. Zellmann, D. Seifried, N. Morrical, I. Wald, W. Usher, J. P. Law-Smith, S. Walch-Gassner, and A. Hinkenjann. Point containment queries on ray-tracing cores for amr flow visualization. Computing in Science & Engineering, 24(02):40–51, March 2022.

[186] Jingke Zhang, Jing Liu, Wei Fan, Weibao Qiu, and Jianwen Luo. Partial hadamard encoded synthetic transmit aperture for high frame rate imaging with minimal l 2-norm least squares method. Physics in Medicine & Biology, 67(10):1–18, 2022.

[187] Xiaoqian Zhang, Maolin Luo, Zhaodi Wen, Qin Feng, Shengshi Pang, Weiqi Luo, and Xiaoqi Zhou. Direct fidelity estimation of quantum states using machine learning. Phys. Rev. Lett., 127:130503, Sep 2021.

[188] K. Zhou, E. Zhang, J. Bittner, and P. Wonka. Visibility-driven mesh analysis and visualization through graph cuts. IEEE transactions on visualization and computer graphics, 14:1667–74, 11 2008.

[189] X. Zou, S. B. Lo, R. Sevilla, O. Hassan, and K. Morgan. The generation of 3D surface meshes for NURBS-enhanced FEM. Computer-Aided Design, 168:103653, 2024.