Intrinsic approaches to learning and computing on curved surfaces

Wiersma, R.T.

DOI
10.4233/uuid:63932f79-d73d-45db-8cec-ce450204b939

Publication date
2024

Document Version
Final published version

Citation (APA)
Wiersma, R. T. (2024). *Intrinsic approaches to learning and computing on curved surfaces*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:63932f79-d73d-45db-8cec-ce450204b939

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Intrinsic approaches to learning and computing on curved surfaces

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended publicly on Tuesday 15 October 2024 at 12:30.

by

Ruben Timotheüs WIERSMA

Master of Science in Computer Science, Delft University of Technology, the Netherlands
born in Leiden, the Netherlands.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. E. Eisemann,	Technische Universiteit Delft, promotor
Prof. dr. J. Dik,	Technische Universiteit Delft, promotor
Independent members:	
Prof. dr. O. Sorkine-Hornung,	Eidgenössische Technische Hochschule Zürich
Prof. dr. M.T.J. Spaan,	Technische Universiteit Delft
Prof. dr. ir. C. Vuik,	Technische Universiteit Delft
Prof. dr. T. Weyrich,	Friedrich-Alexander-Unversität Erlangen-Nürnberg
Other members:	
Dr. K.A. Hildebrandt,	Technische Universiteit Delft

Dr. K.A. Hildebrandt has contributed greatly to the preparation of this dissertation.

Keywords: Computer Graphics, Geometry Processing, Machine Learning, Geometric Deep Learning, Appearance capture

Printed by: Hetuitgeefhuis

An electronic version of this dissertation is available at
https://repository.tudelft.nl/.

Summary

This dissertation develops intrinsic approaches to learning and computing on curved surfaces. Specifically, we work on three tasks: analyzing 3D shapes using convolutional neural networks (CNNs), solving linear systems on curved surfaces, and recovering appearance properties from curved surfaces using multi-view capture. We argue that we can find more efficient and better performing algorithms for these tasks by using intrinsic geometry.

Chapter two and three consider CNNs on curved surfaces. We would like to find patterns with meaningful directional information, such as edges or corners. On images, it is straightforward to define a convolution operator that encodes directional information, as the pixel grid provides a global reference for directions. Such a global coordinate system is not available for curved surfaces. Chapter two presents *Harmonic Surface Networks*. We apply a 2D kernel to the surface by using local coordinate systems. These local coordinate systems could be rotated in any direction around the normal, which is a problem for consistent pattern recognition. We overcome this ambiguity by computing complex-valued, rotation-equivariant features and transporting these features between coordinate systems with parallel transport along shortest geodesics.

Chapter three presents *DeltaConv*. DeltaConv is a convolution operator based on geometric operators from vector calculus, such as the Laplacian. A benefit of the Laplacian is that it is invariant to local coordinate systems. This solves the problem of a missing global coordinate system. However, the Laplacian operator is also isotropic. That means it cannot pick up on directional information. DeltaConv constructs anisotropic operators by splitting the Laplacian into gradient and divergence and applying a non-linearity in between. The resulting convolution operators are demonstrated on learning tasks for point clouds and achieve state-of-the-art results with a relatively simple architecture.

Chapter four considers solving linear systems on curved surfaces. This is relevant for many applications in geometry processing: smoothing data, simulating or animating 3D shapes, or machine learning on surfaces. A common way to solve large systems on grid-based data is a multigrid method. Multigrid methods require a hierarchy of grids and the operators that map between the levels in the hierarchy. We show that these components can be defined for curved surfaces with irregularly spaced samples using a hierarchy of graph Voronoi diagrams. The resulting approach, *Gravo Multigrid*, achieves solving times comparable to the state-of-the-art, while taking an order of magnitude less time for pre-processing: from minutes to seconds for meshes with over a million vertices.

Chapter five demonstrates the use of intrinsic geometry in the setting of appearance modeling, specifically capturing spatially-varying bidirectional reflectance distribution functions (SVBRDF). A low-cost setup to recover SVBRDFs is to capture photographs from multiple viewpoints. A challenge here, is that some reflectance behavior only shows up under certain viewing positions and lighting conditions, which means that we might not be able to tell one material type from another. We frame this as a question of (un)certainty: how certain are we, based on the input data? We build on previous work that shows that the reflection function can be modeled as a convolution of the BRDF with the incoming light. We propose improvements to the convolution model and develop algorithms for uncertainty analysis fully contained in the frequency domain. The result is a fast and uncertainty-aware SVBRDF recovery on curved surfaces.

Samenvatting

Dit proefschrift ontwikkelt intrinsieke methoden voor algoritmes op gekromde oppervlakken. We behandelen drie taken: analyse van 3D-vormen met convolutionele neurale netwerken (CNN's), stelsels van lineaire vergelijkingen op gekromde oppervlakken en de reconstructie van materiaaleigenschappen op basis van foto's uit verschillende kijkhoeken. We betogen dat gebruik van intrinsieke geometrie efficiëntere en beter presterende algoritmes mogelijk maakt.

Hoofdstuk twee en drie behandelen CNN's op gekromde oppervlakken. We willen patronen herkennen met een richting, zoals randen en hoeken. Voor afbeeldingen is dit eenvoudig, omdat het pixelraster een globaal referentiekader geeft voor richtingen. Zo'n globaal coördinatenstelsel is niet aanwezig op gekromde oppervlakken. Hoofdstuk twee presenteert *Harmonic Surface Networks*. We plaatsen een 2D-kernel op het oppervlak door gebruik te maken van lokale coördinatenstelsels. Deze lokale coördinatenstelsels zouden in elke richting rond de normaal kunnen worden gedraaid, wat een probleem is voor consistente patroonherkenning. We lossen deze ambiguïteit op door *features* te berekenen die equivariant zijn voor draaiingen en als complexe getallen worden opgeslagen. Deze *features* worden tussen coördinatenstelsels verplaatst met behulp van parallel transport langs de kortste geodeten.

Hoofdstuk drie presenteert *DeltaConv*. DeltaConv is een convolutie-operator op basis van geometrische operatoren uit de vectoranalyse, zoals de Laplace-operator. De Laplace-operator verandert niet als lokale coördinatenstelsels veranderen. Dit lost het probleem op van een ontbrekend globaal coördinatenstelsel. Een nadeel hiervan is dat de Laplace-operator isotroop is, wat betekent dat deze geen patronen met richting kan oppikken. DeltaConv splitst de Laplace-operator op in de gradiënt en divergentie en past een non-lineariteit toe tussen beide, waardoor een anisotrope operator wordt verkregen. De methode is getest op de analyse van 3D puntwolken en behaalt competitieve resultaten met een relatief eenvoudige architectuur.

Hoofdstuk vier behandelt stelsels van lineaire vergelijkingen op gekromde oppervlakken. Deze zijn relevant voor vele toepassingen: *smoothing* van data, simuleren en animeren van 3D-vormen, en *machine learning*. De multigrid-methode is een veelvoorkomende manier om grote stelsels van lineaire vergelijkingen op te lossen. Multigrid-methoden vereisen een hiërarchie van rasters en een manier om tussen niveaus te communiceren. We laten zien dat dit mogelijk is voor gekromde oppervlakken met onregelmatig verdeelde punten via een hiërarchie van Voronoi-diagrammen op grafen. Onze methode, *Gravo Multigrid*, behaalt oplostijden die vergelijkbaar zijn met de competitie, terwijl het een ordegrootte minder tijd nodig heeft voor de voorbereidende berekeningen: van minuten naar seconden voor meshes met meer dan een miljoen knooppunten.

Hoofdstuk vijf demonstreert het gebruik van intrinsieke geometrie in de context van materiaaleigenschappen, het vastleggen van *spatially-varying bidirectional reflectance distribution functions* (SVBRDF). Een laagdrempelige manier om SVBRDF's te meten is met foto's uit verschillende kijkhoeken. Sommige reflecties zijn echter alleen te zien onder bepaalde kijkhoeken en lichtomstandigheden, waardoor we sommige materialen moeilijk kunnen onderscheiden onder bepaalde omstandigheden. We formuleren dit als een kwestie van (on)zekerheid: hoe zeker zijn we op basis van de data? Eerder werk toont aan dat de reflectiefunctie kan worden gezien als een convolutie van de BRDF met het inkomende licht. We stellen verbeteringen voor aan dit model en ontwikkelen algoritmen voor onzekerheidsanalyse die volledig in het frequentiedomein plaatsvinden. Het resultaat is een snelle SVBRDF-reconstructie met informatie over onzekerheid.

Contents

Introduction 1

Figure 1.1: Triangles covering a curved landscape. Original: "*Blick von Warmbrunn auf die Kleine Sturmhaube*" by Caspar David Friedrich (1810).

Of all the topics he could have picked, Professor Gauss chose geometry. Bernhard Riemann was preparing for his habilitation dissertation, the final examination for the qualification to teach at the university of Göttingen. It was tradition for the candidate to propose three potential topics for this important talk. Riemann had chosen two topics he had already worked on and, as a third topic, geometry. It was expected that the advisor would pick the first, familiar, topic. It must have been a remarkable pick for Gauss to choose the third one. Riemann was said to have had a minor breakdown, but quickly recouped, presenting his habilitation dissertation *'Über die Hypothesen, welche der Geometrie zu Grunde liegen'* (On the Hypotheses which lie at the Foundations of Geometry) on June 10, 1854. Even more remarkable is that the impact of the habilitation dissertation would stretch far beyond the lecture rooms of Göttingen, directly influencing Albert Einstein's general theory of relativity.

This telling is based on the history by Michael Spivak in Volume 2, Chapter 4 of Spivak [214], p. 132-133.

It comes as no surprise that Gauss had an interest in geometry. Gauss had been involved in a number of geodesy surveys years before. The surveyors would measure distances and angles between key points on the landscape, tracing out triangles across the map. These triangles then form a web of key points, distances, and angles, straddling the surface of the land (Figure 1.1). It was of interest to understand the curvature of these triangles. For example, the triangles on a mountain peak form a sharp cone. A flat stretch of farmland has the triangles laying in a planar disc. Gauss had studied ways to measure such curvature in his landmark 1827 paper, titled 'General Investigations of Curved Surfaces.' In this paper, Gauss describes a way to measure curvature (Gaussian curvature) and derives the *Theorema Egregium*, the remarkable theorem.

The *Theorema Egregium* states that the Gaussian curvature of a surface can be derived from measurements taken from within the surface, intrinsic measurements: distances and angles and how they change on the surface. This is remarkable, because he did not expect to get such rich information from only intrinsic measurements. Going back to the geodesy survey, this means that one can use the distances and angles between key points to know the curvature of the land. Gauss himself connects the dots in his 1827 paper and provides some example computations based on a geodesy survey he had performed himself. On a larger scale, it means that you could conclude that the earth is round, without ever leaving the surface of the earth. You could simply take measurements of angles and distances on a curved triangle drawn over the surface of the earth. One step further, on a more abstract level, Gauss' theory allows us to study surfaces that are impossible to realize in physical, three-dimensional space, such as hyperbolic surfaces and, as we will see later, four-dimensional spacetime. Truly remarkable, indeed.

In the abstract of his paper on curvature, Gauss drew the following consequence regarding relations (e.g., distances between points) on surfaces, relevant to this dissertation:

> *We see that two essentially different relations must be distinguished, namely, on the one hand, those that presuppose a definite form of the surface in space; on the other hand, those that are independent of the various forms which the surface may assume.*

In other words, some relationships depend on how a shape sits in space, such as the coordinates in an external frame of reference. If you transform the shape, the coordinates will transform as well. In this dissertation, we will refer to this information as *extrinsic* geometry. Other relationships do not change when the surface is transformed without stretching it, such as distances or angles. Imagine a piece of cloth that is folded and bent, but not stretched. The distances along the cloth do not change as you manipulate it. We will refer to this information as *intrinsic* geometry.

Let us return the story of Riemann, before continuing to the contributions of the current dissertation. In his habilitation dissertation, Riemann picked up where Gauss left off. His first point of business was to define spaces and surfaces in higher dimensions. Gauss had studied surfaces in 3D space, which makes sense, given our experience of the space around us. But what if you wanted to add other dimensions, such as colour? Surfaces with higher dimensions would be named manifolds and curvature on these manifolds could also be defined using intrinsic measurements, such as distances and angles. The crucial question is how these distances should be measured. What is the analog of a ruler that we can use to measure distance in an abstract space, such as colour? Riemann describes

how to do this with infinitesimally short line-elements and we call this measurement 'device' the *metric tensor*. In short, Riemann gave us the tools to study curvature of higher dimensional surfaces (manifolds) intrinsically. Musing on the implications of these findings, Riemann states that

> *this leads us into the domain of another science, of physics, of which the object of this work does not allow us to go today.*

Einstein took over the baton and applied Riemann's theory in physics. He realized that the dimension of time is linked with the three dimensions of space, resulting in a four-dimensional spacetime. Even though we are three-dimensional beings, we can study the curvature of four-dimensional spacetime by taking intrinsic measurements (*Theorema Egregium*). In this way, he was able to predict verifiable measurements that show that spacetime is curved, providing an explanation for gravity: the general theory of relativity.

The purpose of this history is to illustrate the power of using intrinsic geometry to study the world around us. In this dissertation, we seek to apply this perspective in learning and computing on curved surfaces: the organic forms of human bodies and digital characters, the slight bends and sharp corners of household appliances, or a sphere representing lighting and viewing directions. We focus on tasks where we require intrinsic geometric information and want to be robust to changes in extrinsic geometry, such as varying human poses. In the words of Gauss, we want to use "those [properties] that are independent of the various forms which the surface may assume." By using intrinsic geometry, we can find more efficient and better performing algorithms for complex, real-world problems. This is possible, because we ignore details that are irrelevant or counterproductive to the problem at hand.

While intrinsic approaches could simplify our theoretical analysis, they are challenging to implement in practice. Typical signal-processing algorithms work with signals discretized in grids: equally spaced samples on a flat domain. For example, images are processed on a pixel grid. In contrast, we need to deal with irregularly-spaced samples on a curved domain to formulate intrinsic algorithms for data on curved surfaces. The irregularity of the samples complicate implementations of typical procedures like convolution, signal reconstruction, and up- or down-sampling. Curvature means that the metric (how distances and angles are measured) varies across the surface and that we have no global coordinate system to work with. This introduces a number of problems that we highlight in this dissertation.

1.1 CNNs on Curved Surfaces

In Chapter two and three, we consider convolutional neural networks on curved surfaces. Convolutional neural networks (CNNs) are used in machine learning to analyze and synthesize signals that exhibit translation-invariant patterns. CNNs are used, for example, to classify and segment images [127] and as a building block in denoising diffusion models to generate images [94]. In a CNN, a localized kernel with learned weights is convolved over the input image. The benefit of using convolution is that the weights are shared across the input image. This requires fewer parameters (more efficient learning) and the operations are translation invariant (better generalizability).

On images, it is straightforward to define a convolution operator, as we can assume that each pixel is laid out on a grid. This makes it simple to implement convolution using 2D arrays and implies a global coordinate system to align the kernel to. Consider a kernel that detects edges. We can exploit the fact that each pixel is aligned to the same pixel grid to find vertical and horizontal edges using a vertical and horizontal edge-kernel. On surfaces, points are not laid out on a pixel grid, but spaced irregularly. There is no global coordinate system for tangential directions. Yet, we would like to find patterns with meaningful directional information, such as edges and corners. Chapter two and three present solutions to this problem from two different perspectives.

In chapter two, we present *Harmonic Surface Networks*. We apply a 2D kernel to the surface by using a local parametrization: features on the curved surface are mapped to the tangent plane using the exponential map. The exponential map tells us for each point on the surface where it lands on the tangent plane by flattening a local patch of surface. The lack of a global coordinate system for curved surfaces shows up in this setting, because the kernel could be rotated in any direction around the normal. Rather than choosing one direction ad-hoc, we accept this ambiguity and locally resolve mismatches in direction between different points. This is achieved by using rotation-equivariant kernels and transporting the resulting complex-valued features between coordinate systems without changing their direction (parallel transport) along shortest geodesics.

In Chapter three, we present *DeltaConv*. In this chapter, we interpret convolution through the lens of geometric operators. An example of the connection between convolution and operators is the heat equation, $\partial y / \partial t = \Delta y$, which describes the change of a function y over time t to the Laplacian operator applied to y. For any time t, this equation can be solved by convolving y with a heat kernel. The Laplacian has been applied in neural networks for surfaces and graphs, for example in GCN [112] and DiffusionNet [201]. A benefit of the Laplacian to define convolution, is that

the Laplacian is invariant to local coordinate systems. This solves the problem of a missing global coordinate system. However, the Laplacian operator is also isotropic; it does not distinguish between different directions. That means it cannot pick up on directional information. With DeltaConv, we show how to learn anisotropic operators. We split the Laplacian into gradient and divergence and apply a non-linearity in between. We demonstrate the resulting convolution operators on learning tasks for point clouds and show that we can achieve state-of-the-art results with a relatively simple architecture.

1.2 Geometric Multigrid

In Chapter four, we shift our focus to solving linear systems on curved surfaces. Such linear systems show up in many geometry-processing tasks: smoothing data, simulating or animating 3D shapes, or machine learning on surfaces. The size of these linear systems depends on the resolution of the surface discretization, which could run into millions of vertices or points. A common way to solve such large systems on grid-based data is a multigrid method, where the linear system is solved on multiple grids in a hierarchy. The two main components of a multigrid method are the hierarchy of grids and the operators that map between the levels in the hierarchy. Our challenge lies in defining both these components for a curved surface with irregularly spaced samples. We show that this can be achieved with a hierarchy of graph Voronoi diagrams. The resulting approach, called *Gravo Multigrid*, achieves solving times comparable to the state-of-the-art, while taking an order of magnitude less time for pre-processing: from minutes to seconds for meshes with over a million vertices.

1.3 Material appearance capture

In the penultimate chapter of this dissertation, we transition out of the geometry processing territory toward material- and appearance capture. We show that the strategy of using intrinsic geometry for solving complex real-world problems can also be applied in this setting.

Object acquisition is the task of capturing the geometry and appearance of a physical object. The resulting digital copy of the object can be used to render the objects in virtual settings, for example, the original setting of a cultural-heritage object or a scene in an animated movie. For many objects, it suffices to model how light interacts with the object at the surface. We can model how light reflects off the surface using a bidirectional reflectance distribution function (BRDF). When this function varies over the surface, a spatially-varying BRDF (SVBRDF) is used. A low-cost setup to

recover the SVBRDF is to capture a number of photographs from multiple viewpoints and to capture the lighting environment. However, even with hundreds of photographs, one might not be able to recover the parameters of the SVBRDF at every point. This is because some reflectance behavior only shows up under certain viewing positions and lighting conditions. How can we know when we have the right conditions for capture?

In Chapter five, we provide insight into these questions by transitioning from the spatial domain to the frequency domain. We build on previous work that shows that the reflection function can be modeled as a convolution of the BRDF with the incoming light [190]. If the incoming- and outgoing light are known, the task of finding the BRDF can be seen as a deconvolution. This signal processing perspective can help us answer if we have the right signal available for recovery. We propose improvements to the convolution model and propose algorithms for uncertainty analysis fully contained in the frequency domain. The result is a fast and uncertainty-aware SVBRDF recovery on curved surfaces.

1.4 Curved surfaces, convolution, and least-squares

Even though the chapters in this dissertation span a wide range of application domains, we soon get familiar with a recurring cast of characters. Each of the chapters considers a problem defined on a curved domain; we aim to use convolution on this domain; and use least-squares as a way to map concepts defined in a continuous setting to the discrete setting. The reason for this recurrence is the underlying strategy to use the intrinsic geometry of a problem to generate simpler and better algorithms. For CNNs on surfaces and geometric multigrid methods, the problem can be reduced to a 2D curved surface embedded in 3D space. For reflectance, the problem can be reduced to the sphere of incoming and outgoing directions. Both are examples of *curved surfaces*. We simplify the problems by using *convolution*: hereby, our solutions become *invariant* to translations and rotations. Finally, it is challenging to create algorithms that work on irregular and sparse discretizations. We find this is the case for meshes and point clouds, but also for samples of a radiance field from a sparse set of viewpoints. Our algorithms are robust to these sparse and irregular samples by using a *least-squares* approach and fitting regularization.

In the following chapters, we further detail these efforts. Each chapter provides a detailed introduction to the problem context and related work, a description of the method, and experiments demonstrating the claimed benefits. The dissertation is concluded with chapter seven, where we reflect on the lessons learned and challenges for future work.

CNNs on Surfaces using Rotation-Equivariant Features

2

*This chapter * is concerned with a fundamental problem in geometric deep learning that arises in the construction of convolutional neural networks on surfaces. Due to curvature, the transport of filter kernels on surfaces results in a rotational ambiguity, which prevents a uniform alignment of these kernels on the surface. We propose a network architecture for surfaces that consists of vector-valued, rotation-equivariant features. The equivariance property makes it possible to locally align features, which were computed in arbitrary coordinate systems, when aggregating features in a convolution layer. The resulting network is agnostic to the choices of coordinate systems for the tangent spaces on the surface. We implement our approach for triangle meshes. Based on circular harmonic functions, we introduce convolution filters for meshes that are rotation-equivariant at the discrete level. We evaluate the resulting networks on shape correspondence and shape classifications tasks and compare their performance to other approaches.*

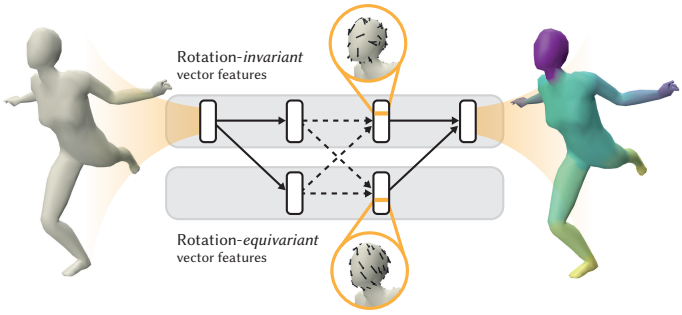


Figure 2.1: We propose CNNs on surfaces that operate on vectors and separate rotation-equivariant and rotation-invariant features.

* This chapter is based on the paper "CNNs on Surfaces using Rotation-Equivariant Features" published in ACM Transactions on Graphics (SIGGRAPH 2020) [244].

2.1 Introduction

The success of Deep Learning approaches based on convolutional neural networks (CNNs) in computer vision and image processing has motivated the development of analogous approaches for the analysis, processing, and synthesis of surfaces. Along these lines, approaches have been proposed for problems such as shape recognition [218], shape matching [13], shape segmentation [152], shape completion [138], curvature estimation [79], and 3D-face synthesis [191].

In contrast to images, which are described by regular grids in a Euclidean domain, surfaces are curved manifolds and the grids on these surfaces are irregular. In order to still use regular grids, one can work with multiple projections of the surface on planes [218] or with volumetric grids [252].

An alternative to learning on regular grids is generalized deep learning, often referred to as geometric deep learning [23], which targets irregularly sampled manifolds and graphs. A central element of such geometric CNNs is a generalized convolution operator. For CNNs on images, the convolution layers are built from convolution kernels, which are transported across the image. As a result, the parameters that define one kernel describe the convolution across the whole image, which significantly reduces the number of parameters that need to be learned. This is a motivation for exploring constructions of generalized convolution operators on surfaces based on convolution kernels.

To apply a convolution kernel defined on $\mathbb{R}^2$ to a function at a point on a surface, the Riemannian exponential map is used to locally lift the function from the surface to a function defined on the tangent plane at the point. By identifying the tangent plane with $\mathbb{R}^2$, the convolution of the kernel and the lifted function can be computed. In this way, the convolution kernel can be applied everywhere on the surface. However, a problem arises, since there is a rotational degree of freedom when $\mathbb{R}^2$ is identified with a tangent plane. Moreover, the transport of filters on a surface depends on the chosen path. If a filter is transported along two different ways from one point of a surface to another, the transported filters are rotated against each other. This *rotation ambiguity* problem is fundamental and caused by the curvature of the surface.

The rotation ambiguity problem can be addressed by specifying a coordinate system at each point of the surface, *e.g.* according to the principal curvature directions [13, 163, 178] or the direction of maximum activation [155, 221]. As a consequence, however, the coordinate systems in the local neighborhoods are arranged in different patterns for each point. For a network this means that, when features are aggregated to form the next layer of the network, the features are not only dependent on the sequence of

convolution kernels that are applied, but also on the arrangement of coordinate systems in the local neighborhoods. Loosely speaking, the information contained in the features in the neighborhood of a point can be arbitrarily rotated against the coordinate system at the point. One can think of this as in a cubist painting, where the elements that make up a structure, for example the eyes, nose, and mouth on a face, are rotated against each other.

Multi-Directional Geodesic CNNs (MDGCNN) [184] provide an alternative approach to the rotation ambiguity problem. The idea is to sample the rotational degree of freedom regularly, to compute the convolutions for each of the sample directions, and to feed the results for all directions into the network. The multi-directional features are pooled at the final layer. The disadvantage of this approach is that each filter must not only be evaluated once at each point, but also for rotated versions of the filter and the results need to be stored. To build an operable network, the filters are only evaluated in some sample directions and the results are linearly interpolated to get results in intermediate directions, introducing inaccuracies in each consecutive layer.

We introduce a novel network architecture that does not suffer from the rotation ambiguity problem. The features in this network are rotation-equivariant and organized in streams of different equivariance classes (Figure 2.1). These streams interact with each other in the network and are finally merged into a rotation-invariant output. The resulting network is independent of the choice of coordinate systems in the tangent spaces, which means that it does not suffer from the rotation ambiguity problem. To realize this network, we work with vector-valued, instead of scalar-valued, features and introduce rotation-equivariant convolution and pooling operators on meshes. The convolution operators use the Riemannian exponential map and parallel transport to convolve vector-valued kernels on $\mathbb{R}^2$ with tangent vector fields on meshes.

As kernels on $\mathbb{R}^2$ we use the circular harmonics, which are known to be rotation-equivariant. We prove that the resulting discrete convolution operators on meshes are equivariant with respect to rotations of the coordinate system in the tangent spaces. Due to the rotation-equivariance property, it suffices to compute the convolution at each point with respect to an arbitrary reference coordinate system in the tangent plane. Then, if the result with respect to any other coordinate system is required, one only needs to transform the result of the convolution in the reference coordinate system to the other coordinate system. The rotation-equivariance property is still valid if several of these filters are applied consecutively, *e.g.* in the deeper layers of a network. Rotation-equivariance enables the vector-valued convolution operator to always align the features in a local neighborhood of a point to the coordinate system at the point. Our network architecture builds upon Harmonic Nets [247], which

are used for rotation-equivariant learning on images. Therefore, we call the networks *Harmonic Surface Networks* (HSN).

We implement the Harmonic Surface Networks for triangle meshes. Based on circular harmonic filters, we derive discrete convolution filters that are equivariant with respect to basis transformations in the tangent spaces associated to the vertices of the mesh. The parameters of the filters separate radial and angular direction, which leads to a low number of parameters for each kernel, when compared to other filter kernels for surface meshes. We experimentally analyze the properties and performance of the HSNs and compare the performance to other geometric CNNs for surface meshes.

In summary, our main contributions are:

- We introduce Harmonic Surface Networks, which combine a vector-valued convolution operation on surfaces and rotation-equivariant filter kernels to provide a solution to the rotation ambiguity problem of geometric deep learning on surfaces.
- Based on circular harmonics, we derive convolution filters for meshes that have the desired rotational-equivariance properties at the discrete level, and, additionally, allow for separating learning of parameters in radial and angular direction.
- We analyze the Harmonic Surface Networks for surface meshes and compare the performance to alternative approaches.

2.2 Related work

In this section, we provide a brief overview of work on geometric deep learning closely related to our approach. For a comprehensive survey on the topic, we refer to [23].

Charting-based methods Closest to our work are so-called *charting-based methods*. For convolution, these methods use local parametrizations to apply filter kernels to features defined on the surface. A seminal approach in this direction are the Geodesic CNNs (GCNN, [155]). They consider a parametric model of convolution filters in $\mathbb{R}^2$ using Gaussian kernels placed on a regular polar grid. For convolution, the filters are mapped to the surface using the Riemannian exponential map. Other intrinsic methods use anisotropic heat kernels [13], learn the shape parameters of the kernels [163], learn not only the parameters of the kernels but also pseudo-coordinates of the local parametrizations [230], or use B-Splines [64] or Zernike polynomials [221] instead of Gaussians.

A challenging problem for these constructions is the lack of canonical coordinate systems on a surface, inducing the rotation ambiguity problem, discussed in the introduction. To address the

ambiguity, maximum pooling over the responses to a sample of different rotations can be used [155, 221] or the coordinate systems can be aligned to the (smoothed) principal curvature directions [13, 163, 178]. Both approaches have their downsides. Angular pooling discards directional information and the alignment to principal curvature directions can be difficult as the principal curvature directions are not defined at umbilic points and can be unstable in regions around umbilic points. A sphere, for example, consists only of umbilic points. A solution to this problem are the Multi-Directional Geodesic CNNs (MDGCNN) [184]. At every point, the filters are evaluated with respect to multiple choices of coordinate systems. This can be formalized by describing the features by so-called directional functions instead of regular functions. Parallel transport is used to locally align the directional functions for convolution.

Our approach provides a different solution to the rotation ambiguity problem that does not require computing the filters in multiple directions and storing the results. Once the filter in one direction is computed, the rotation-equivariance property of our filters allows us to obtain the results for the other directions by applying a rotation. We compare our approach to MDGCNN in Section 2.5. Parallel Transport Vector Convolution [198] is a convolution operation for vector fields on manifolds that was used to learn from vector valued data on surfaces. Similar to this approach, we also use parallel transport to define convolution for vector fields on surface. A concept for the construction of Gauge-equivariant convolution filters on manifolds is introduced in [32] along with a concrete realization of a Gauge CNN for the icosahedron. In recent work, an adaption of Gauge CNNs to surface meshes was introduced [40].

Instead of local charts, one can also parametrize an area globally and then learn on the parameter domain [152, 84]. An advantage of this approach is that standard CNNs can be applied to the parameter domain. A disadvantage is that global parametrizations lead to larger metric distortion than local parametrizations. In addition, typically different global parametrization methods are needed for surfaces of different genus.

Spectral methods and graph CNNs An alternative to charting-based methods are spectral methods. For images, CNNs can operate in the Fourier domain. Spectral Networks [24] generalize CNNs to graphs using the spectrum of a graph Laplacian. To reduce the computational complexity, ChebNet [41] and Graph Convolutional Networks (GCN) [112] use local filters based on the graph Laplacian. Recently, various approaches for defining CNNs on graphs have been introduced, we refer to [273, 251] for recent surveys. This line of work diverges from our approach since we specialize to discrete structures that describe two-dimensional manifolds. Furthermore, we aim at analyzing the surface underlying a mesh, which means

our method aims to be agnostic of the connectivity of the mesh, *i.e.* the graph underlying a mesh. The concept of local filtering has also been applied to surfaces using the Laplace–Beltrami and Dirac operator [117]. MeshCNN [85] generalizes graph CNNs to mesh structures by defining convolution and pooling operations for triangle meshes.

Point clouds PointNet [186] is an approach for creating CNNs for unordered sets of points. First, a neighborhood is constructed around each point with a radius ball. To retrieve a response for point i, a function is applied to each neighbor of i and the maximum activation across i's neighbors is stored as the new response for i. PointNet++ [187] is an extension of PointNet with hierarchical functionality. Because PointNet applies the same function to each neighbor, it is effectively rotation-invariant. DGCNN [236] extends PointNet++ by dynamically constructing neighborhood graphs within the network. This allows the network to construct and learn from semantic neighborhoods, instead of mere spatial neighborhoods. PointCNN [132] learns a χ-transformation from the point cloud and applies the convolutions to the transformed points. TextureNet [98] uses 4-rotational symmetric fields to define the convolution domains and applies operators that are invariant to 4-fold symmetries. While we evaluate our approach on meshes, our method can be used to process point clouds sampled from a surface.

Symmetric spaces Specialized approaches for symmetric surfaces such as the sphere [31, 115] have been proposed. On the one hand, the approaches profit from the highly symmetric structure of the surfaces, while on the other hand they are limited to data defined on these surfaces.

Rotation-equivariance Our approach builds on Harmonic Networks [247]. This work is part of a larger effort to create group-equivariant networks. Different approaches such as steerable filters [67, 142, 34, 241], hard-baking transformations in CNNs [33, 151, 61, 124], and learning generalized transformations [93] have been explored. Most relevant to our approach are steerable filters, since we use features that can be transformed, or steered, with parallel transport. The core idea of steerable filters is described by [67] and applied to learning by [142]. The key ingredient for these steerable filters is to constrain them to the family of circular harmonics. [247] added a rotation offset and multi-stream architecture to develop Harmonic Networks. The filters in Harmonic Networks are designed in the continuous domain and mapped to a discrete setting using interpolation. Harmonic Networks was built on by [225] for Tensor Field Networks. Tensor Field Networks achieve rotation- and translation-equivariance for 3D point clouds by moving from the

family of circular harmonics to that of spherical harmonics. In doing so, they lose the phase offset parameter, as it is not commutative in SO(3). Spherical harmonics were also used for rigid motion-invariant processing of point clouds [185].

2.3 Background

Harmonic Networks Harmonic Nets (H-Nets) [247] are rotation-equivariant networks that can be used to solve computer vision tasks, such as image classification or segmentation, in such a way that a rotation of the input does not affect the output of the network. H-Nets restrict their filters to circular harmonics, resulting in the following filter definition:

$$W_m(r, \theta, R, \beta) = R(r)e^{i(m\theta+\beta)}, \tag{2.1}$$

where r and θ are polar coordinates, $R : \mathbb{R}_+ \to \mathbb{R}$ is the *radial profile*, $\beta \in [0, 2\pi)$ is a *phase offset*, and $m \in \mathbb{Z}$ is the rotation order. The cross-correlation of W_m with a complex function x at a point p is given by the integral

$$[W_m \star x](p) = \int_0^{\epsilon} \int_0^{2\pi} R(r)e^{i(m\theta+\beta)}x(r, \theta)\, r\, d\theta\, dr. \tag{2.2}$$

This filter is rotation-equivariant with respect to rotations of the domain of the input to the filter:

$$[W_m \star x^{\phi}](p) = e^{im\phi}[W_m \star x^0](p), \tag{2.3}$$

where $x^0(r, \theta)$ is a complex function and $x^{\phi}(r, \theta) = x(r, \theta - \phi)$ is the same function defined on a rotated domain. The rotation order m determines how the output of the filters changes when the domain of the input is rotated. For $m = 0$, the result does not change and the filter is rotation invariant. For $m \geq 1$ the result is rotated by an angle that is m times the original angle, which we refer to as m-equivariance.

An important property of these filters is that, if filters of orders m_1 and m_2 are chained, rotation-equivariance of order $m_1 + m_2$ is obtained:

$$[W_{m_1} \star [W_{m_2} \star x^{\phi}] = e^{im_1\phi}e^{im_2\phi}[W_{m_1} \star [W_{m_2} \star x^0] \tag{2.4}$$

$$= e^{i(m_1+m_2)\phi}[W_{m_1} \star [W_{m_2} \star x^0]. \tag{2.5}$$

This is integral to the rotation-equivariance property of the network as a whole. The network architecture of H-Nets is structured in separate streams per rotation order as illustrated in Figure 2.2. For each feature map inside a stream of order M, we require that the sum of rotation orders m_i along any path reaching that

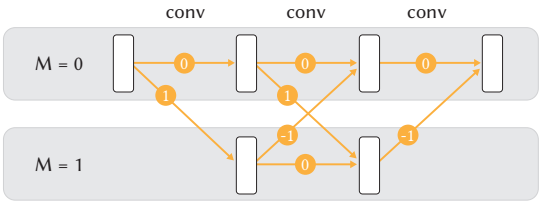


Figure 2.2: Harmonic Networks separate the result of different rotation order convolutions into streams of M-equivariance.

feature map equals M. In the last layer of an H-Net, the streams are fused to have the same rotation order. The rotation order of the output stream determines the class of equivariance of the network and is chosen to match the demands of the task at hand. For rotation invariant tasks, the last layer has order $m = 0$. Still, in the hidden layers of the network, rotation-equivariant streams capture and process information that is not rotation-invariant, which yields improved performance compared to networks built only from rotation-invariant filters.

Riemannian exponential map Let v be a vector in the tangent plane $T_p\mathcal{S}$ at a point p of a surface $\mathcal{S}$. Then, there is exactly one geodesic curve starting at p in direction v. If we follow this geodesic curve until we have covered a length that equals the norm of v, we end up at a point q on the surface. The Riemannian exponential map associates each vector v in $T_p\mathcal{S}$ to the corresponding point q on the surface. This map is suitable as a local parametrization of the surface, because it is a local diffeomorphism, *i.e.*, bijective and smooth with smooth inverse. Furthermore, the mapping is an isometry in radial direction away from p. The construction of the Riemannian exponential map and its inverse, the logarithmic map, is illustrated in Figure 2.3. An example of the Riemannian exponential map is the azimuthal projection of the sphere, which is used in cartography and is included in the emblem of the United Nations.

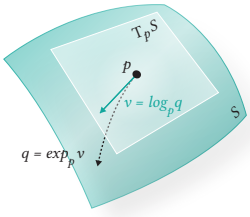


Figure 2.3: The Riemannian exponential- and logarithmic map.

In graphics, the Riemannian exponential map is used, for example, for texture decalling [196] and shape modeling [197]. In geometric deep learning [23], it is used to map filter kernels defined on the tangent plane to filters defined on the surface and to lift functions defined on the surface to functions defined on the tangent plane. Recent approaches for computing the Riemannian exponential map on surface meshes are based on heat diffusion [204, 90]. These methods reduce the computation of the exponential map to solving a few sparse linear systems and can be accurately computed globally. Alternatives are approaches based on Dijkstra's algorithm [196, 159].

Parallel transport of vectors In the Euclidean plane one can describe vector fields by specifying x and y coordinates relative to a global coordinate system. There is no such coordinate system on

curved surfaces. To be able to compare vectors at neighboring points p and q of a surface, we use the parallel transport along the shortest geodesic curve c connecting the points. If v is a tangent vector at p which has the angle α with the tangent vector of c at p, then the vector transported to q is the tangent vector that has an angle of α with the tangent of c at q and has the same length as v.

To describe tangent spaces on meshes and to compute the parallel transport, we use the Vector Heat Method [204], which we also use to calculate the Riemannian exponential map.

2.4 Method

In this section, we describe the building blocks of Harmonic Surface Networks. We start by introducing notation and then discuss convolutions, linearities, non-linearities, and pooling. Finally, we discuss why the networks provide a solution to the rotation ambiguity problem by analyzing how the choices of coordinate systems in the tangent spaces affect the convolution operations and HSNs.

Notation The features in an HSN are associated to the vertices (or just a subset of the vertices) of a triangle mesh. To each vertex, we assign a coordinate system in the tangent plane at the vertex and use complex numbers to represent tangent vectors with respect to the coordinate system. We denote the feature vector of rotation order M in network layer l at vertex i by $\mathbf{x}_{i,M}^{l}$. To simplify the notation, we leave out the indices l and M when they are not relevant. For HSN, these feature vectors are complex-valued. Every operation applied to these vectors is performed element-wise. For example, when we multiply a feature vector with a complex number, each component of the vector is multiplied by this number. We use parallel transport along the shortest geodesic from vertex j to vertex i to transport the feature vectors. The transport depends on the geometry of the mesh, the chosen coordinate systems at the vertices, and the rotation order of the feature. It amounts to a rotation of the features. In practice, we store the rotation as a complex number with the angle of rotation ϕ_{ji} as its argument and apply rotations to vectors via complex multiplication:

$$P_{j\to i}(\mathbf{x}_{j,M}) = e^{i(M\phi_{ji})}\mathbf{x}_{j,M}. \tag{2.6}$$

For every vertex, we construct a parametrization of a local neighborhood of the vertex over the tangent plane using the Riemannian logarithmic map. We represent the map by storing polar coordinates r_{ij} and θ_{ij} with respect to the coordinate system in the tangent plane for every vertex j in the neighborhood of vertex i.

Convolution The convolution layers of HSNs combine the rotation-equivariance of circular harmonics with the transport of features along surfaces. We use compactly supported filter kernels to limit the number of neighboring features that are used in a convolutional layer. Let $\mathcal{N}_i$ denote the set of vertices that contribute to the convolution at i. In the case of continuous features on a Euclidean space, the correlation with W_m is given by an integral, see (2.2). For discrete meshes, we approximate the integral with a sum and use parallel transport and the Riemannian logarithmic map to evaluate the filter:

$$\mathbf{x}_{i,M+m}^{(l+1)} = \sum_{j \in \mathcal{N}_i} w_j \left(R(r_{ij}) e^{i(m\theta_{ij}+\beta)} P_{j \to i}(\mathbf{x}_{j,M}^{(l)}) \right), \tag{2.7}$$

where r_{ij} and θ_{ij} are the polar coordinates of point j in $T_i\mathcal{S}$, and $P_{j \to i}$ is the transport of features defined in (2.6). The integration weights w_j are

$$w_j = \frac{1}{3} \sum_{jkl} A_{jkl} \tag{2.8}$$

where A_{jkl} is the area of triangle jkl and the sum runs over all triangles of the mesh containing vertex j. The weights can be derived from numerical integration with piecewise linear finite elements, see [239].

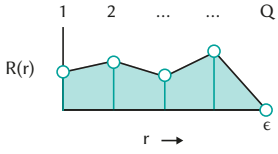


Figure 2.4: We parametrize the radial profile by learning the value at Q equally spaced rings and linearly interpolating for values in between.

The radial profile R and the phase offset β are the learned parameters of the filter. To represent the radial profile, we learn values at equally spaced locations in the interval $[0, \epsilon]$, where ϵ provides a bound on the support of the filter. To get a continuous radial profile, the learned values are linearly interpolated as illustrated in Figure 2.4. At a point with radial coordinate r_{ij}, the radial profile is

$$R(r_{ij}) = \sum_{q}^{Q} \mu_q(r_{ij}) \rho_q, \tag{2.9}$$

where $\mu_q(r_{ij})$ is a linear interpolation weight and ρ_q a learned weight matrix. We set $\rho_Q = 0$, which bounds the support of the kernel.

To speed up training, we precompute three components: the integration weight, the interpolation weight to each radial profile point, and the rotation by the angle θ_{ij}:

$$\text{precomp}_{ij} = \left(w_j \mu_q(r_{ij}) e^{im\theta_{ij}} \right). \tag{2.10}$$

We precompute the polar coordinates and integration weights with the Vector Heat method [204]. The precomputation is stored in a [Q x 2] matrix for each (i, j) pair.

Linearities Linearities are applied to complex features by applying the same linearity to both the real and imaginary component,

resulting in a linear combination of the complex features.

Non-Linearities We follow Harmonic Networks [247] for complex non-linearities: non-linearities are applied to the radial component of the complex features, in order to maintain the rotation-equivariance property. In our experiments, we used a complex version of ReLU

$$\mathbb{C}\text{-ReLU}_b(\mathbf{X}e^{i\theta}) = \text{ReLU}(\mathbf{X} + b)e^{i\theta}, \tag{2.11}$$

where b is a learned bias added to the radial component. If the non-linearity were to be applied without bias, it would be an identity operation, as the radius is always positive.

Pooling and unpooling Pooling layers downsample the input by aggregating regions to representative points. We need to define an aggregation operation suitable for surfaces and a way to choose representative points and create regions. We use farthest point sampling and cluster all non-sampled points to sampled points using geodesic nearest neighbors.

The aggregation step in pooling is performed with parallel transport, since the complex features of points within a pooling region do not exist in the same coordinate system. Thus, we define the aggregation step for representative point i with points j in its pooling cluster $\mathcal{C}_i$ as follows:

$$\mathbf{x}_{i,M} = \square_{j \in \mathcal{C}_i} P_{j \to i}(\mathbf{x}_{j,M}), \tag{2.12}$$

where $\square$ is any aggregation operation, such as sum, max, or mean. In our implementation, we use mean pooling. Pooling happens *within* each rotation order stream, hence the rotation order identifier M for both $\mathbf{x}_i$ and $\mathbf{x}_j$.

The sampling of points per pooling level and construction of corresponding kernel supports are performed as a precomputation step, so we can compute the logarithmic map and parallel transport for each pooling level in precomputation.

Unpooling layers upsample the input by propagating the features from the points sampled in the pooling layer to their nearest neighbors. Parallel transport is again applied to align coordinate systems.

Rotation orders To maintain rotation-equivariance throughout the network, the output of the filters is separated in streams of rotation orders. The output of a filter applied to $\mathbf{x}_{j,M}$ with rotation order m should end up in rotation order stream $M' = M + m$. The output from two convolutions resulting in the same stream is summed. For example, a convolution on $\mathbf{x}_{j,1}$ with $m = -1$ and a convolution on $\mathbf{x}_{j,0}$ with $m = 0$ both end up in the stream $M' = 0$ and are summed. A visual overview can be found in Figure 2.6, on the right. We only

apply parallel transport to inputs from the $M > 0$ rotation-order streams, as the values in the $M = 0$ stream are rotation-invariant.

Properties of HSNs In the following, we show that the result of convolutions (2.7) as well as the layers of HSNs commute with transformations of the coordinate systems in the tangent spaces. This means that we can choose any coordinate system in the tangent spaces and use it to build, train, and evaluate our networks. A different choice of the coordinate systems leads to the same network, only with transformed features.

As a consequence, HSNs do not suffer from the rotation ambiguity problem, which we described in the introduction. The rotation ambiguity problem is caused by the fact that due to the curvature of a surface, there is no consistent choice of coordinate systems on a surface. So if a filter that is defined on $\mathbb{R}^2$ is to be applied everywhere on the surface, coordinate systems in the tangent planes must be defined. Since there is no canonical choice of coordinate systems, the coordinate systems at pairs of neighboring points are not aligned. Due to the commutativity property (Lemma 2.4.1), a convolution layer of our HSNs can use the features that are computed in arbitrary coordinate systems and locally align them when computing the convolutions.

In contrast, a network without that property cannot locally align the features. If one would want to align the coordinate systems' features locally, they would have to recompute the features starting from the first layer. Since this is not feasible, one needs to work with non-aligned coordinate systems. The result is that the same kernel is applied with different coordinate systems at each location. Moreover, as features are combined in later layers, the network will learn from local relations of coordinate systems. This is undesirable, as these relations hold no meaningful information: they arise from arbitrary circumstances and choices of coordinate systems.

To prove that the features of HSNs commute with the coordinate changes, this property must be shown for the individual operations. The non-linearity is invariant to changes of the basis as it only operates on the radial coordinates. Here, we restrict ourselves to convolution, as for pooling, one can proceed analogous to this proof.

Lemma 2.4.1 *The convolution (2.7) commutes with changes of coordinate systems in the tangent planes.*

Proof. We represent the coordinate system of a tangent space by specifying the x-axis. Coordinates of points are then given by a complex number. If we rotate the coordinate system at vertex i by an angle of $-\phi$, the features of order M transform to

$$\mathbf{x}_{i,M} = e^{iM\phi}\mathbf{x}_{i,M}, \tag{2.13}$$

where $\mathbf{x}_{i,M}$ is the feature in the rotated coordinate system. The change of coordinate system also affects the polar coordinates of any vertex j in the tangent plane of i and the transport of features from any vertex j to i

$$\tilde{\theta}_{ij} = \theta_{ij} + \phi \qquad \tilde{P}_{j\to i}(\mathbf{x}_{j,M}) = e^{iM\phi} P_{j\to i}(\mathbf{x}_{j,M}), \tag{2.14}$$

where $\tilde{\theta}_{ij}$ is the angular coordinate of j in the tangent plane of i with respect to the rotated coordinate system and $\tilde{P}$ is the transport of features with respect to the rotated coordinate system. Then, convolution with respect to the rotated coordinate system is given by

$$\mathbf{x}^{(l+1)}_{i,M+m} = \sum_{j\in\mathcal{N}_i} w_j \left(R(r_{ij}) e^{i(m\tilde{\theta}_{ij}+\beta)} \tilde{P}_{j\to i}(\mathbf{x}^{(l)}_{j,M}) \right). \tag{2.15}$$

Plugging (2.14) into (2.15), we get

$$\mathbf{x}^{(l+1)}_{i,M+m} = \sum_{j\in\mathcal{N}_i} w_j \left(R(r_{ij}) e^{im\phi} e^{i(m\theta_{ij}+\beta)} e^{iM\phi} P_{j\to i}(\mathbf{x}^{(l)}_{j,M}) \right) \tag{2.16}$$

$$= e^{i(M+m)\phi} \mathbf{x}^{(l+1)}_{i,M+m}. \tag{2.17}$$

The latter agrees with the basis transformations of the features, see (2.13). If the coordinate system at a vertex j that is neighboring i is rotated by some angle, then this affects the transport of features $P_{j\to i}$ from j to i and the feature $\mathbf{x}^{(l)}_{j,M}$. However, since the transport and feature are rotated by the same angle but in opposite directions, the term $P_{j\to i}(\mathbf{x}^{(l)}_{j,M})$ is not affected. □

We visualize the convolution of a simple $m = 0$ feature with $m = 0$ and $m = 1$ filters in Figure 2.5. For each example, $\beta = 0$ and $R(r) = 1 - r$. The inputs are basic patterns: a) a vertical edge, b) a diagonal edge, c) two vertical edges, and d) two horizontal edges. We observe that $m = 0$ convolutions smooth the signal and $m = 1$ convolutions activate on edge boundaries. We can also see Equation 2.13 at work: the input in a) and b) differs by a 45°-rotation of the domain. The $m = 0$ outputs therefore are related by the same rotation of the domain. The equivariant $m = 1$ outputs are related by a rotation of the domain and an additional rotation of the features. The same holds for c) and d), now with a rotation by 90°.

Discretization In this chapter, we detail Harmonic Surface Networks for meshes, as they are sparse and convenient representations of surfaces. Yet, we have attempted to formulate the building blocks for HSNs as general as possible. Thus, one could replace the use of the words 'mesh' and 'vertex' with 'point cloud' and 'point' and the method would remain largely the same. The main differences would

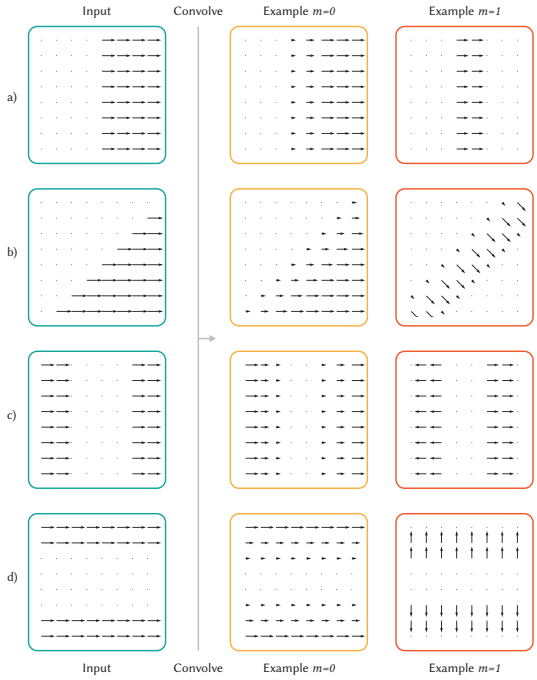


Figure 2.5: Examples of simple input vector fields (one feature) and the response of a convolution with $m = 0$ and $m = 1$ kernels.

be (1) how to compute the logarithmic map and parallel transport and (2) how to define a weighting per point. The first question has been answered by [204]: the Vector Heat method can compute a logarithmic map for any discretization of a surface. The latter can be resolved with an appropriate integral approximation scheme.

2.5 Experiments

The goal of our experiments is twofold: we will substantiate our claims about improved performance with comparisons against state-of-the-art methods and we aim to evaluate properties of HSNs in isolation, in particular the benefits of the multi-stream architecture and rotation-equivariance.

2.5.1 Implementation

In the following paragraphs we discuss the architecture and data processing used in our experiments.

Following recent works in geometric deep learning [230, 85, 184], we employ a multi-scale architecture with residual connections. More specifically, we recreate the deep U-ResNet architecture from [184]. The U-ResNet architecture consists of four stacks of ResNet blocks, organized in a U-Net architecture with pooling and unpool-

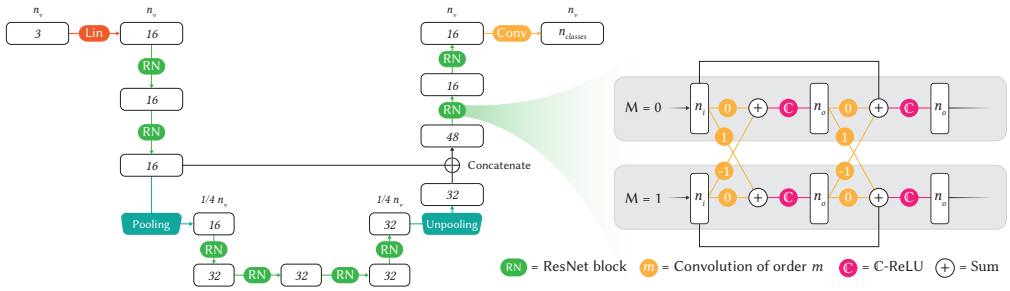


Figure 2.6: HSN U-ResNet architecture used for correspondence and shape segmentation. On on the left, the full architecture; on the right, a detail of the ResNet Block.

ing layers (Figure 2.6). Each stack consists of two ResNet blocks, which, in turn, consist of two convolutional layers with a residual connection. We use 16 features in the first scale and 32 features in the second scale. Finally, we configure our network with two rotation order streams: $M = 0$ and $M = 1$, learning a rotation-equivariant kernel for every connection between streams.

For pooling operations, we sample a quarter of the points on the surface using farthest point sampling and take the average of the nearest neighbors. These neighbors are computed with the Heat Method [36], by diffusing indices from sampled points to all other points with a short diffusion time ($t = 0.0001$). With each pooling step, we grow the filter support by $1/\sqrt{\text{ratio}}$. Thus, we grow the radius with a factor 2. At the unpooling stage, we propagate the features from the points sampled in the pooling stage to the nearest neighbors, using parallel transport. We use ADAM [111] to minimize the negative log-likelihood and train for 100 epochs on correspondence and 50 epochs for shape segmentation and mesh classification.

For each experiment, we normalize the scale of each shape, such that the total surface area is equal to 1. We then compute local supports for each vertex i by assigning all points within a geodesic disc of radius ϵ to its neighborhood $\mathcal{N}_i$. We normalize the weighting described in Equation 2.8 by the sum of weights for each neighborhood, to compensate for the different sizes of support of the filters in the different layers, in particular, after pooling and unpooling.

Next, we employ the Vector Heat Method out of the box [204, 203] to compute the logarithmic map and parallel transport for each neighborhood. For our multi-scale architecture, we first compute local supports for each scale and coalesce the resulting graph structures into one multi-scale graph. This way, we can precompute the necessary logarithmic maps in one pass. Wherever we can, we use raw xyz-coordinates as input to our network. To increase the robustness of the network against transformations of the test set, we randomly scale and rotate each shape with a factor sampled from

$\mathcal{U}(0.85, 1.15)$ and $\mathcal{U}(-\frac{1}{8}\pi, \frac{1}{8}\pi)$, respectively. For correspondence, we use SHOT descriptors, to provide a clean comparison against previous methods.

The networks are implemented using PyTorch Geometric [63]. The implementation can be retrieved from `https://github.com/rubenwiersma/hsn`.

2.5.2 Comparisons

Following from the benefits outlined in the introduction, we expect HSNs to show improved results over existing spatial methods, even though fewer parameters are used for each kernel. We experimentally validate these expected benefits by applying HSN on three tasks: shape classification on SHREC [135], correspondence on FAUST [12], and shape segmentation on the human segmentation dataset proposed by [152].

Table 2.1: Results for shape classification on 10 training samples per class of HSN against previous work.

Method	Accuracy
HSN (ours)	**96.1%**
MeshCNN	91.0%
GWCNN	90.3%
GI	88.6%
MDGCNN	82.2%
GCNN	73.9%
SG	62.6%
ACNN	60.8%
SN	52.7%

Shape classification We train an HSN to perform classification on the SHREC dataset [135], consisting of 30 classes (Figure 2.7). We only train on 10 training samples per class. Like [85], we take multiple random samplings from the dataset to create these training sets and average over the results. We train for 50 epochs, compared to the 200 epochs used in previous works. This task is challenging due to the low number of training samples and labels. Therefore, we reduce the size of our network and consequently, the number of parameters to learn. We only use the first half of the U-ResNet architecture, with only one ResNet block per scale. To obtain a classification, we retrieve the radial components from the last convolutional layer, followed by a global mean pool. The initial radius of our filters is $\epsilon = 0.2$ and the number of rings in the radial profile is 6.

For our comparisons, we cite the results from [85] and [58], comparing our method to MeshCNN [85], SG [21], SN [252], GI [210], and GWCNN [58]. Additionally, we train MDGCNN [184], GCNN [155], and ACNN [13] with the exact same architecture as

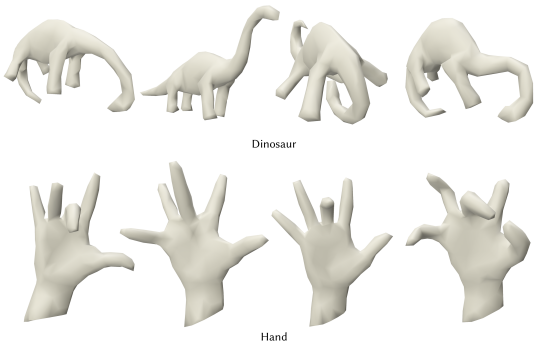


Figure 2.7: Two example classes with four shapes each from the SHREC shape classification dataset.

HSN. The results are outlined in Table 2.1. HSN outperforms all previous methods, demonstrating the effectiveness of our approach, even for lower training times. One explanation for the large gap in performance is the low number of training samples. HSN uses fewer parameters, resulting in fewer required training samples. This is supported by results in [247], who also show higher performance for small datasets compared to other methods. The low number of samples also explains why some non-learning methods outperform learning methods. An additional problem faced by ACNN is the low quality of the meshes, resulting in a disparate principal curvature field. This obstructs the network from correctly relating features at neighboring locations and degrades performance. This same effect is observed when applying HSN without parallel transport, aligned to principal curvature directions (Table 2.5).

Shape segmentation Next, we demonstrate HSN on shape segmentation. We train our network to predict a body-part annotation for each point on the mesh. We evaluate our method on the dataset proposed by Maron et al. [152], which consists of shapes from FAUST ($n = 100$) [12], SCAPE ($n = 71$) [7], Adobe Mixamo ($n = 41$) [1], and MIT ($n = 169$) [232] for training and shapes from the human category of SHREC07 ($n = 18$) for testing. The variety in sources provides a variety in mesh structures as well, which tests HSN's robustness to changes in mesh structure.

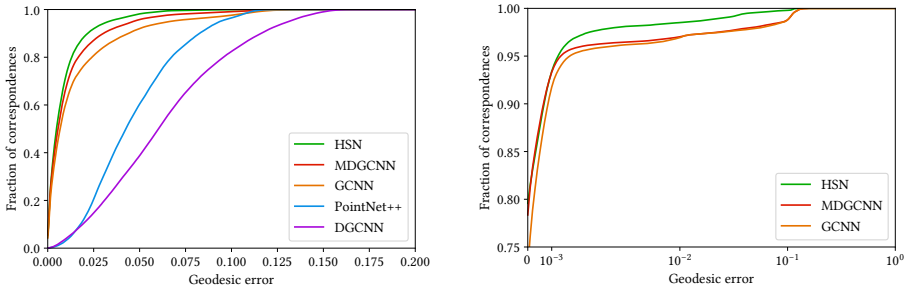
We use the U-ResNet architecture, providing xyz-coordinates as input and evaluate the class prediction directly from the final convolutional layer. The logarithmic map and parallel transport are computed using the original meshes from [152]. To limit the training time, 1024 vertices are sampled on each mesh using farthest point sampling to be used in training and testing. This type of downsampling is also applied by [85] (to 750 vertices) for similar reasons. The initial support of our kernels is $\epsilon = 0.2$ and the number of rings in the radial profile is 6.

We report the accuracy as the fraction of vertices that was classified correctly across all test shapes. For the comparisons, we cite the results from [184], [84] and [85]. HSN produces competitive results compared to state-of-the-art methods, performing only slightly worse than MeshCNN.

Table 2.2: Results for shape segmentation by HSN and related methods.

Method	# Features	Accuracy
HSN (ours)	3	91.14%
MeshCNN	5	**92.30%**
SNGC	3	91.02%
PointNet++	3	90.77%
MDGCNN	64	89.47%
Toric Cover	26	88.00%
DynGraphCNN	64	86.40%
GCNN	64	86.40%
ACNN	3	83.66%

Figure 2.8: Vector-valued featuremap and label predictions from our Harmonic Surface Network trained on shape segmentation.



(a) Fraction of correspondences for a given geodesic error on the remeshed FAUST dataset using HSN (ours), MDGCNN, GCNN, PointNet++, and DGCNN.

(b) Fraction of correspondences for a given geodesic error on the original FAUST dataset using HSN (ours), MDGCNN, and GCNN. Note: the x axis is set to logarithmic scale.

To understand the complex features learned by the network, we visualize the features on the model, alongside our predictions for segmentation. We can interpret the complex features as intrinsic vectors on the surface and visualize them as such using Polyscope [200]. Figure 2.8 shows a single feature in the $M = 1$ stream from the second-to-last layer. We observe high activations on certain bodyparts (legs, hands) and note that the intrinsic vectors are aligned. Our interpretation is that the corresponding filter 'detects' legs and hands. The alignment of features is beneficial to the propagation of these activations through the network; if features are oriented in opposing directions, they can cancel each other out when summed.

Correspondence Correspondence finds matching points between two similar shapes. We evaluate correspondence on the widely used FAUST dataset [12]. FAUST consists of 100 scans of human bodies in 10 different poses with ground-truth correspondences. We set up the network to predict the index of the corresponding vertex on the ground-truth shape. To this end, we add two fully connected layers (FC256, FCN_{vert}) after the U-ResNet architecture. The initial radius of our filters is $\epsilon = 0.1$ and the number of rings in the radial profile is 2.

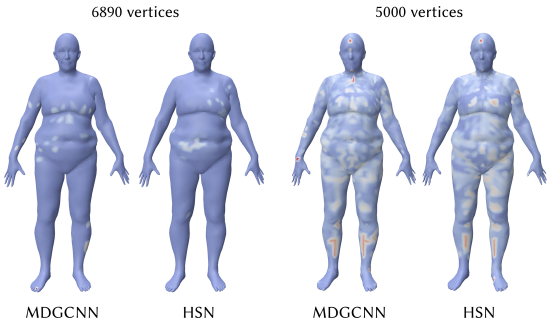


Figure 2.10: Geodesic error visualised on the test shapes, shown: the first test shape for MDGCNN and HSN, for both the original and remeshed dataset.

We train on the first 80 shapes and report the fraction of correct correspondences for a given geodesic error on the last 20 models (Figures 2.9a and 2.9b). As input to our network, we provide 64-dimensional SHOT descriptors, computed for 12% of the shape area. Similar to Poulenard and Ovsjanikov [184], we train our network on a remeshed version of the FAUST dataset as well, since the original FAUST dataset exhibits the same connectivity between shapes. The remeshed dataset is a challenge that is more representative of real applications, where we cannot make any assumptions about the connectivity or discretization of our input. Having recreated the same architecture, with the same input features, we can fairly compare our method to MDGCNN and report the results obtained by Poulenard and Ovsjanikov [184].

The results in Figures 2.9a, 2.9b, and 2.10 show that HSN outperforms the compared state-of-the-art methods. More importantly, HSN improves existing results on the remeshed FAUST dataset, demonstrating the robustness of the method to changes in the discretization of the surface.

Parameter count and memory usage The following calculation shows that HSN achieves this performance using fewer parameters and with less impact on memory during computation. Let n_i and n_o be the number of input and output features, n_ρ and n_θ the number of rings and angles on the polar grid, and n_m the number of rotation order streams. MDGCNN [184] and GCNN [155] learn a weight matrix for every location on a polar grid, resulting in a parameter complexity of $O(n_i\, n_o\, n_\rho\, n_\theta)$. HSN learns the same weight matrix, but only for the radial profile and the phase offset. We chose to learn these weights separately for every connection between streams, resulting in a parameter complexity of $O(n_i\, n_o\, (n_\rho + 1)\, n_m^2)$. If one chooses to learn weights per stream, which the original H-Nets opt for, this is reduced to $O(n_i\, n_o\, (n_\rho + 1)\, n_m)$. Removing the dependency on n_θ has a high impact on the number of parameters: for $n_\rho = 2$ and $n_\theta = 8$, HSN uses 75% of the parameters used by MDGCNN [184] (122880 vs. 163840), only considering the convolutional layers. If we were to use the same number of rings and angles as used by GCNN [155], $n_\rho = 5$ and $n_\theta = 16$, HSN would use only 30% of the parameters used by other spatial methods.

Concerning space complexity, MDGCNN [184] stores the result from every rotation throughout the network, multiplying the memory complexity of the model by the number of directions used, which tends to be between 8 and 16. In comparison, HSN's complex-valued features only increase the memory consumption for storing the separate streams and complex features. For two streams, this increases the space complexity by a factor of 4. This is important for the ability to scale our method to higher-resolution shapes and larger datasets, necessary for use in applications.

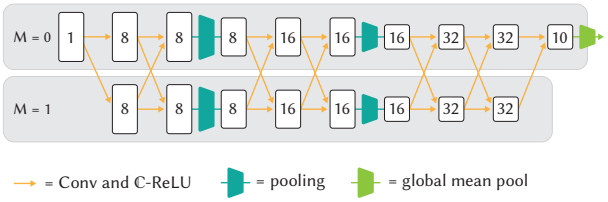


Figure 2.11: Architecture for classification of Rotated MNIST.

Figure 2.12: Rotated MNIST mapped to a sphere

2.5.3 Evaluation

Aside from comparisons with other methods, we intend to get a deeper understanding of the properties of HSN; specifically, the benefit of the $M = 1$ stream w.r.t. rotation-invariant methods and a single $M = 0$ stream.

To evaluate the benefit of different rotation order streams in our method, we compare two different stream configurations of our method: a single-stream architecture ($M = 0$) and a double-stream architecture ($M = 0$ and $M = 1$). We limit this experiment to only these two configurations, because (1) experiments from [247] demonstrate that rotation order streams of $M > 1$ do not significantly improve the performance of the network and (2) the $M = 0$ and $M = 1$ stream features have an intuitive interpretation on the surface as scalar values and intrinsic vectors, respectively.

We evaluate the two configurations on a new task: classification of digits from the Rotated MNIST mapped to a sphere, as well as the shape segmentation and classification tasks from the comparison experiments.

Table 2.3: Results of HSN tested on Rotated MNIST mapped to a sphere for a single- and double-stream configuration.

Method	Streams	Accuracy
HSN	0, 1	**94.10%**
HSN	0	70.68%
HSN (param x4)	0	75.57%

Rotated MNIST on a sphere. We use an elliptical mapping [66] to map the grayscale values from the images of the Rotated MNIST dataset [125] to the vertices of a unit sphere with 642 vertices. The Rotated MNIST dataset consists of 10000 randomly rotated training samples, 2000 validation samples, and 50000 test images separated in 10 classes. We use an architecture similar to the one in [247]: Conv8, Conv8, Pool0.5, Conv16, Conv16, Pool0.25, Conv32, Conv32, Conv10 (Figure 2.11). The kernel supports for each scale are the following: 0.3, 0.45, 0.8 (the geodesic diameter of the unit sphere is π).

The two stream architecture demonstrates significant improvement over the single-stream architecture with a higher parameter count to compensate for using only one stream: 94.10% vs. 75.57%

(Table 2.3). This affirms the benefit of rotation-equivariant streams for learning signals on surfaces.

Shape Segmentation and Classification We repeat our experiments for shape segmentation and classification in four different configurations: the double-stream configuration used in section 5.2; a single-stream configuration; a single-stream configuration with four times the parameters; and the double stream configuration aligned to a smoothed principal curvature field, instead of local alignment with parallel transport. The single-stream configurations aim to provide insight into the benefit of the rotation-equivariant stream and to rule out the sheer increase in parameters as the cause of this performance boost. The last configuration shows the benefit of locally aligning rotation-equivariant features over aligning kernels to smoothed principal curvature fields, as is done by ACNN [13].

The results in Table 2.4 and Table 2.5 show that the double-stream architecture improves the performance from satisfactory to competitive. Furthermore, an increase in parameters has a negative impact on performance, likely due to the relatively low number of training samples. This becomes even more apparent when comparing the learning curves for each configuration in Figure 2.13: the double-stream configuration is more stable and performs better than both single-stream configurations. These results support the benefit of the rotation-equivariant stream.

Finally, we find that HSN performs significantly better when using parallel transport than when aligned with a smoothed principal curvature direction (pc aligned): for shape segmentation, the benefit introduced by the $M = 1$ stream is diminished, and for shape classification, we observe a large drop in performance, likely induced by the coarseness of the meshes and the resulting low-quality principal curvature fields.

Table 2.4: Results of HSN tested on shape segmentation for multiple configurations.

Method	Streams	Accuracy
HSN	0, 1	**91.14%**
HSN	0	88.74%
HSN (param ×4)	0	87.25%
HSN (pc aligned)	0, 1	86.22%

Table 2.5: Results of HSN tested on classification for multiple configurations.

Method	Streams	Accuracy
HSN	0, 1	**96.1%**
HSN	0	86.1%
HSN (pc aligned)	0, 1	49.7%

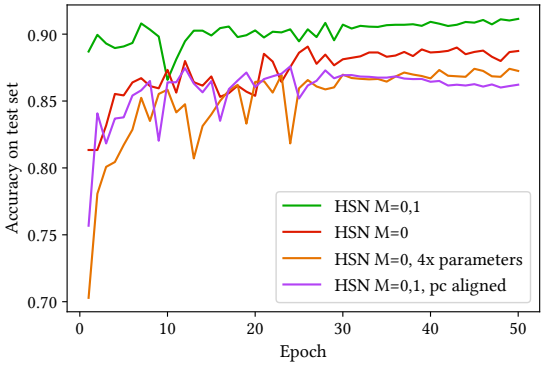


Figure 2.13: Validation accuracy per training epoch several configurations of HSN on shape segmentation.

2.6 Conclusion

We introduce Harmonic Surface Networks, an approach for deep learning on surfaces operating on vector-valued, rotation-equivariant features. This is achieved by learning circular harmonic kernels and separating features in streams of different equivariance classes. The advantage of our approach is that the rotational degree of freedom, arising when a filter kernel is transported along a surface, has no effect on the network. The filters can be evaluated in arbitrarily chosen coordinate systems. Due to the rotation-equivariance of the filters, changes in the coordinate system can be recovered after the convolutions have been computed by transforming the results of the convolution. The convolution operator uses this property and always locally aligns the features.

We implement this concept for triangle meshes and develop convolution filters that have the desired equivariance properties at the discrete level and allow for separating learning of parameters in the radial and angular direction. We demonstrated in our comparisons that HSNs are able to produce competitive or better results with respect to state-of-the-art approaches and analyzed the benefits of rotation-equivariant streams as an addition to rotation-invariant streams and parallel transport for local alignment compared to global alignment.

While we only implemented our method for meshes, every component of HSNs can be transferred to point clouds. We aim to extend our implementation and evaluate the benefits of HSNs for learning on point clouds.

Another promising direction is the application of outputs from the rotation-equivariant stream. We expect that the ability to learn intrinsic vectors on a surface can facilitate new applications in graphics and animation.

DeltaConv: Anisotropic Operators for Geometric Deep Learning on Point Clouds

3

Learning from 3D point-cloud data has rapidly gained momentum, motivated by the success of deep learning on images and the increased availability of 3D data. In this chapter, we aim to construct anisotropic convolution layers that work directly on the surface derived from a point cloud. This is challenging because of the lack of a global coordinate system for tangential directions on surfaces. We introduce DeltaConv, a convolution layer that combines geometric operators from vector calculus to enable the construction of anisotropic filters on point clouds. Because these operators are defined on scalar- and vector-fields, we separate the network into a scalar- and a vector-stream, which are connected by the operators. The vector stream enables the network to explicitly represent, evaluate, and process directional information. Our convolutions are robust and simple to implement and match or improve on state-of-the-art approaches on several benchmarks, while also speeding up training and inference.*

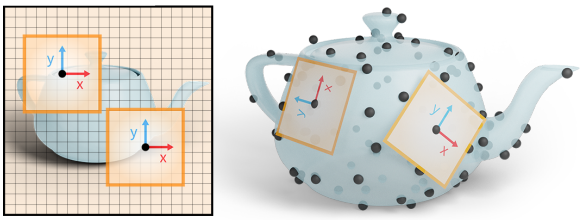


Figure 3.1: Images have a global coordinate system (left). Point clouds do not (right), complicating the design of anisotropic convolutions.

* This chapter is based on the paper "DeltaConv: Anisotropic Operators for Geometric Deep Learning on Point Clouds" published in ACM Transactions on Graphics (SIGGRAPH 2022) [246].

3.1 Introduction

The success of convolutional neural networks (CNNs) on images and the increasing availability of point-cloud data motivate generalizing CNNs from images to 3D point clouds [82, 143, 23]. One way to achieve this is to design convolutions that operate directly on the surface. Such *intrinsic* convolutions reduce the kernel space to tangent spaces, which are two-dimensional on surfaces. Compared to extrinsic convolutions, intrinsic convolutions can be more efficient and the search space for kernels is reduced, they naturally ignore empty space, and they are robust to rigid- and non-rigid deformations [13]. Examples of intrinsic convolutions on point clouds are GCN [112], PointNet++ [187], EdgeConv [236], and DiffusionNet [201].

Our focus is on constructing intrinsic convolutions which are anisotropic or direction-dependent. This is difficult because of the fundamental challenge that non-linear manifolds lack a global coordinate system. As an illustration of the problem, consider a CNN on images (Figure 3.1, left). Because an image has a globally consistent up-direction, the network can build anisotropic filters that activate the same way across the image. For example, one filter can test for vertical edges and the other for horizontal edges. No matter where the edges are in the image, the filter response is consistent. In subsequent layers, the output of these filters can be combined, e.g., to find a corner. Because we do not have a global coordinate system on surfaces (Figure 3.1, right), one cannot build and use anisotropic filters in the same way as on images. This limits current intrinsic convolutions on point clouds. For example, GCNs filters are isotropic. PointNet++ uses maximum aggregation and adds relative point positions, but still applies the same weight matrix to each neighboring point.

We introduce a new way to construct anisotropic convolution layers for geometric CNNs. Our convolutions are described in terms of geometric operators instead of kernels. The operator-based perspective is familiar from GCN, which uses the Laplacian on graphs. While the Laplacian is a natural fit for intrinsic learning on surfaces, it is isotropic. A classical way of creating anisotropic operators is to write the Laplacian as the divergence of the gradient and apply a linear or non-linear operation on the intermediate vector field [240]. We build on this idea by constructing learnable anisotropic operators from elemental geometric operators: the gradient, co-gradient, divergence, curl, Laplacian, and Hodge-Laplacian. These operators are defined on spaces of scalar fields and tangential vector fields. Hence, our networks are split into two streams: one stream contains scalars and the other tangential vectors. The operators map along and between the two streams. The vector stream encodes feature activations and directions along the surface, allowing the network to test and relate directions in subsequent layers. Depending on the

Figure 3.2: A ResNet with varying convolutions is overfitted to a target image created with twenty anisotropic diffusion steps. DeltaConv can reproduce the filter well, where other convolutions struggle. (Courtesy NASA)

task, the network outputs scalars or vectors. A property of a network constructed from these operators is that it is coordinate-independent: though bases of the tangent spaces of a point cloud need to be chosen, the weights learned by the network will be the same no matter what bases are chosen. Hence, we can realize direction-dependent convolutions despite the lack of global coordinate systems on surfaces and without the need of specially constructed tangent space bases. We name our convolutions *DeltaConv*.

To get an idea of the benefits of DeltaConv, consider the anisotropic image filter proposed by Perona and Malik [181]. The Perona–Malik filter integrates an anisotropic diffusion equation in which the anisotropic operator combines the gradient, a non-linearity, and the divergence. DeltaConv has access to the building blocks needed to construct such an anisotropic operator and to perform explicit integration steps of the diffusion equation. This is illustrated in Figure 3.2. We trained a simple ResNet [87] to match the result of twenty anisotropic diffusion steps on a sample image. While DeltaConv can reproduce the filter well, other intrinsic convolutions and regular image convolutions fail to capture the effect, producing overly smooth signals or artifacts instead. Additional benefits of our approach are the following: by maintaining a stream of vector features throughout the network, our convolutions can relate directional information between different points on the surface. Together with the increased expressiveness of convolutions due to anisotropy, this results in increased accuracy over isotropic convolutions, as well as state-of-the-art approaches, as we show in our experiments. Also, each operator is implemented as a sparse matrix and the combination of operators is computed per point, which is simple and efficient.

In our experiments, we demonstrate that a simple architecture with only a few DeltaConv blocks can match and, in some cases, outperform state-of-the-art results using more complex architectures. We achieve 93.8% accuracy on ModelNet40, 84.7% on the most difficult variant of ScanObjectNN, 86.9 mIoU on ShapeNet, and 99.6% on SHREC11, a dataset of non-rigidly deformed shapes. Our

ablation studies show that adding the vector stream can decrease the error by up to 25% (from 90.4% to 92.8%) on ModelNet40 and up to 21% for ShapeNet (from 81.1 to 85.1 mIoU), while the use of per-point directional features speeds up inference by $1.5 - 2\times$ and the backward pass by $2.5 - 30\times$ compared to edge-based features.

Summarizing our main contributions:

- We introduce a new construction of convolution layers for geometric CNNs that supports the construction of anisotropic filters. This is achieved by letting networks learn convolutions as compositions and linear combinations of geometric differential operators and point-wise non-linearities. Moreover, the networks maintain a stream of vector features in addition to the usual stream of scalar features and use the operators to communicate in and between the streams.
- We propose a network architecture that realizes our approach and adapt the differential operators to work effectively in our networks.
- We implement and evaluate the network for point clouds[†] and propose techniques to cope with undersampled regions, noise, and missing information prevalent in point cloud learning.

3.2 Related work

We focus our discussion of related work on the most relevant topics. Please refer to surveys on geometric deep learning [23, 22] and point cloud learning [82, 143] for a more comprehensive overview of this expanding field.

Point cloud networks and anisotropy A common approach for learning on point-cloud data is to learn features for each point using a multi-layer perceptron (MLP), followed by local or global aggregation. Many methods also learn features on local point pairs before maximum aggregation. Well-known examples are PointNet and its successor PointNet++ [186, 187]. Several follow-up works improve speed and accuracy, for example by adding more combinations of point-pair features [270, 220, 261, 126, 146, 188, 256, 149]. Some of these point-wise MLPs explicitly encode anisotropy by splitting up the MLP for each 3D axis [123, 146]. Concepts from transformers [228] have also made their way to point clouds [271, 264, 137]. These networks use self-attention to compute aggregation weights for (neighboring) points. Spatial information is incorporated by adding relative positions in 3D. Attention-based aggregation could be used in our approach as a replacement of maximum aggregation. The distance between points could serve as an intrinsic spatial encoding.

[†] The implementation is available at https://github.com/rubenwiersma/deltaconv.

Pseudo-grid convolutions are a more direct translation of image convolutions to point clouds. Many of these are defined in 3D and thus support anisotropy in 3D coordinates. Several works learn a continuous kernel and apply it to local point-cloud regions [145, 17, 144, 224, 250, 92, 8, 64, 255]. Others learn discrete kernels and map points in local regions to a discrete grid [97, 128, 133, 30, 76]. We go into an orthogonal direction by building intrinsic convolutions, which operate in fewer dimensions and naturally generalize to (non-)rigidly deformed shapes.

Finally, graph-based approaches create a k-nearest neighbor- or radius-graph from the input set and apply graph convolutions [209, 236, 265, 141, 205, 48, 28, 223, 62, 234, 268, 177]. DGCNN [236] introduces the EdgeConv operator and a dynamic graph component, which reconnects the k-nearest neighbor graph inside the network. EdgeConv computes the maximum over feature differences, which allows the network to represent directions in its channels. Channel-wise directions *can* resemble spatial directions if spatial coordinates are provided as input, which is only the case in the first layer for DGCNN. In contrast, our convolutions support anisotropy directly in the operators.

Rotation-equivariant approaches Architectures with two streams and vector-valued features are also used in rotation-equivariant approaches for point clouds and meshes. A group of works studies rotation-equivariance in 3D space, aiming to design networks invariant to rigid point-cloud transformations [57, 225, 31, 185]. This concept is also incorporated in the transformer setups [68]. Rotation-equivariant kernels typically output vector-valued features. Vector Neurons simplify their use by linearly combining 3D vectors, followed by a vector non-linearity [42]. Our use of vector MLPs is similar. Differences are that we use tangential vectors, rather than 3D vectors, and we derive these vectors inside the network using geometric operators.

An alternative approach is to build networks using intrinsic rotation-equivariant convolutions on meshes [40, 244, 32, 184, 242, 72]. These networks use local parametrizations and apply rotation- or gauge-equivariant kernels in the parameter domain to achieve independence from the choice of bases in the tangent spaces. Our approach is an alternative to gauge-equivariant networks. The use of differential operators also makes our networks independent of the choice of local coordinate systems. A benefit of our approach is that local parametrizations are not needed. For example, gauge-equivariant approaches typically use the exponential map for local parametrization but neglect the angular distortion induced by the parametrization. To the best of our knowledge, we are the first to implement and evaluate an intrinsic two-stream architecture on point clouds.

Geometric operators Multiple authors use geometric operators to construct convolutions. The graph-Laplacian is used in GCN [112]. Spectral networks for learning on graphs are based on the eigenpairs of the graph-Laplacian [24]. Surface networks for triangle meshes [117] interleave the Laplacian with the extrinsic Dirac operator [139]. Parametrized Differential Operators (PDOs) [105] use the gradient and divergence operators to learn from spherical signals on unstructured grids. DiffGCN [54] uses finite difference schemes of the gradient and divergence operators for the construction of graph networks. DiffusionNet [201] learns diffusion using the Laplace–Beltrami operator and directional features from gradients. DeltaConv uses a larger set of operators, combining and concatenating operators from vector calculus. In addition, it allows the processing of directional information in the stream of vector-valued features. A related approach is HodgeNet [212], which learns to build operators using the structure of differential operators. Outside of deep learning, differential operators are widely applied for the analysis of 3D shapes [35, 39].

3.3 Method

We construct anisotropic convolutions by learning combinations of geometric differential operators. Because these operators are defined on scalar- and vector fields, we split our network into scalar and vector features. In this section, we describe these two streams, the operators and how they are discretized, and how combinations of the operators are learned. Finally, we consider the properties that result from this construction.

Streams Consider a point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$ with N points arranged in an $N \times 3$ matrix. All points can be associated with C additional features, which are stored in a matrix $\mathbf{X} \in \mathbb{R}^{N\times C}$. Inside the network, we refer to the features in layer l at point i as $\mathbf{x}_i^{(l)} \in \mathbb{R}^{C_l}$. All of these features constitute the *scalar stream*.

The *vector stream* runs alongside the scalar stream. Each feature in the vector stream is a tangent vector, encoded by coefficients (α_i^u, α_i^v) with respect to a basis in the corresponding tangent plane. The basis can be any pair of orthonormal vectors that are orthogonal to the normal vector. The coefficients are interleaved for each point, forming the matrix of features $\mathbf{V}^{(l)} \in \mathbb{R}^{2N\times C_l}$. One channel in $\mathbf{V}^{(l)}$ is a column of coefficients: $[\alpha_1^u, \alpha_1^v, \ldots, \alpha_i^u, \alpha_i^v, \ldots, \alpha_N^u, \alpha_N^v]^\intercal$. The input for the vector stream is a vector field defined at each point. In our experiments, we use the gradients of the input to the scalar stream. We will refer to the continuous counterparts of $\mathbf{X}$ and $\mathbf{V}$ as X and V, respectively.

3.3.1 Scalar to scalar: maximum aggregation

A simplified version of point-based MLPs is applied inside the scalar stream, building on PointNet++ [187] and EdgeConv [236]. We apply an MLP per point and then perform maximum aggregation over a k-nn neighborhood $\mathcal{N}(i)$. The features in the scalar stream are computed as

$$\mathbf{x}_i^{(l+1)} = h_{\mathbf{\Theta_0}}(\mathbf{x}_i^{(l)}) + \max_{j \in \mathcal{N}(i)} h_{\mathbf{\Theta_1}}(\mathbf{x}_j^{(l)}), \tag{3.1}$$

where $h_{\mathbf{\Theta_0}}$ and $h_{\mathbf{\Theta_1}}$ denote multi-layer perceptrons (MLPs), consisting of fully connected layers, batch normalization [101], and non-linearities. If point positions are used as input, they are centralized before maximum aggregation: $\mathbf{p}_j = \mathbf{p}_j - \mathbf{p}_i$.

The biggest difference with EdgeConv and PointNet++ is that we use only point-based features within the network instead of edge-based features. The matrix multiplication used inside the MLP is thus not applied to kN feature vectors, but N point-wise feature vectors. This has a significant impact on the run time of the forward and backward passes. Directional information is encoded in per-point vectors instead of edges.

3.3.2 Scalar to vector: Gradient and co-gradient

The gradient and co-gradient operators connect the scalar stream to the vector stream. The gradients of a function represent the largest rate of change and the directions of that change as a vector at each point. The co-gradients are 90-degree rotations of the gradients. Combined, the gradients and co-gradients span the tangent planes, allowing the network to scale, skew, and rotate the gradient vectors.

We construct a discrete gradient operator using a moving least-squares approach on neighborhoods with k neighbors [167]. This approach is used in modeling and processing for point clouds and solving differential equations on point clouds [36, 136]. The procedure and accompanying theory is outlined in the supplemental material. The gradient operator is represented as a sparse matrix $\mathbf{G} \in \mathbb{R}^{2N \times N}$. It takes N values representing features on the points and outputs $2N$ values representing the gradient expressed in coefficients of the tangent basis of each point. The matrix is highly sparse as it only contains $2k$ elements in each row. The co-gradient $\mathbf{JG}$ is a composition of the gradient with a block-diagonal sparse matrix $\mathbf{J} \in \mathbb{R}^{2N \times 2N}$, where each block in $\mathbf{J}$ is a 2×2 90-degree rotation matrix.

Point clouds typically contain undersampled regions and noise. This can be problematic for the moving least-squares procedure. Consider the example in Figure 3.3, a chair with thin legs. Only a few points lie along the line constituting the legs of the chair. Hence, the perpendicular direction to the line is undersampled, resulting

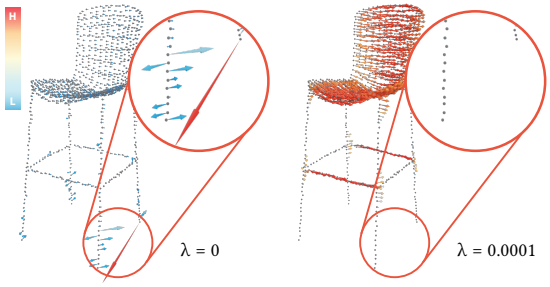


Figure 3.3: Gradient of the x-coordinate on a chair without regularization (left) and with regularization (right).

in a volatile least-squares fit: a minor perturbation of one of the points can heavily influence the outcome (left, circled area). We add a regularization term scaled by λ to the least-squares fitting procedure, which seeks to mitigate this effect (right). This is a known technique referred to as ridge regression or Tikhonov regularization.

We also argue that the gradient operator should be normalized, motivated by how information is fused in the network. If $\mathbf{G}$ exhibits diverging or converging behavior, features resulting from $\mathbf{G}$ will also diverge or converge. This is undesirable when the gradient is applied multiple times in the network. Features arising from the gradient operation would then have a different order of magnitude which needs to be accounted for by the network weights. Therefore, we normalize $\mathbf{G}$ by the ℓ_∞-operator norm, which provides an upper bound on the scaling behavior of an operator

$$\mathbf{G} = \mathbf{G}/|\mathbf{G}|_\infty, \quad \text{where } |\mathbf{G}|_\infty = \max_i \sum_j |\mathbf{G}_{ij}|. \tag{3.2}$$

3.3.3 Vector to scalar: Divergence, Curl, and Norm

The vector stream connects back to the scalar stream with divergence, curl, and norm. These operators are commonly used to analyze vector fields and indicate features such as sinks, sources, vortices, and the strength of the vector field. The network can use them as building blocks for anisotropic operators.

The discrete divergence is also constructed with a moving least-squares approach, which is described in the supplement. Divergence is represented as a sparse matrix $\mathbf{D} \in \mathbb{R}^{N \times 2N}$, with $2kN$ elements. Curl is derived as $-\mathbf{DJ}$.

3.3.4 Vector to vector: Hodge Laplacian

Vector features are diffused in the vector stream using a combination of the identity $\mathbf{I}$ and the Hodge Laplacian $\mathbf{\Delta}$ of V. Applying the Hodge Laplacian to a vector field V results in another vector field encoding the difference between the vector at each point and its neighbors. The Hodge Laplacian can be formulated as a combination

of grad, div, curl and $\mathcal{J}$ [20]

$$\mathbf{\Delta} = -(\text{grad div } + \mathcal{J} \text{ grad curl }). \tag{3.3}$$

In the discrete setting, we replace each operator with its discrete variant

$$\mathbf{L} = -(\mathbf{GD} - \mathbf{JGDJ}). \tag{3.4}$$

3.3.5 Why these operators?

The operators we use are related to each other in a fundamental way. They form a metric version of the *de Rham complex* of a surface [238]. The following diagram lays out the connections described in the previous sections, where each of the operators maps between functions (scalar fields) and vector fields.

$$X \underset{\text{div}}{\overset{\text{grad}}{\rightleftarrows}} V \underset{\text{co-grad}}{\overset{\text{curl}}{\rightleftarrows}} X \tag{3.5}$$

Note that the bottom row is a 90-degree rotated version of the top row. If we follow the diagram from left to right and apply grad and then curl to any function, the output will always be zero. The same holds for the path from right to left. The operators listed are first-order derivatives. Laplacians, which are second-order derivatives, can be formed by composing the first-order operators. For functions: to vector fields with grad and back again with div (Laplace-Beltrami). For vector fields: we go to scalars with div and curl and back again with grad and co-grad (Hodge-Laplacian). DeltaConv learns to combine these operators and supports anisotropy by adding non-linearities in-between.

3.3.6 DeltaConv: Learning Anisotropic Operators

Each of the operations either outputs scalar-valued or vector-valued features. We concatenate all the features belonging to each stream and then combine these features with parametrized functions

$$\begin{aligned}
\mathbf{v}'_i &= \mathbf{h}_{\mathbf{\Theta_0}}(\mathbf{v}_i, (\mathbf{GX})_i, (\mathbf{LV})_i), \\
\mathbf{x}'_i &= h_{\mathbf{\Theta_1}}(\mathbf{x}_i, (\mathbf{DV'})_i, (-\mathbf{DJV'})_i, \|\mathbf{v'}_i\|) + \max_{j \in \mathcal{N}_i} h_{\mathbf{\Theta_2}}(\mathbf{x}_j).
\end{aligned} \tag{3.6}$$

We use the prime to indicate features in layer $l+1$. All other features are from layer l. $h_{\mathbf{\Theta_1}}$ and $h_{\mathbf{\Theta_2}}$ denote standard MLPs. $\mathbf{h}_{\mathbf{\Theta_0}}$ denotes an MLP used for vectors. The vector MLPs scale and sum vectors, which means they do not work on individual vector coefficients and are coordinate-independent. Recall that $\mathbf{V} \in \mathbb{R}^{2N \times C^{(l)}}$ interleaves the vector coefficients for each point in the columns. One layer in

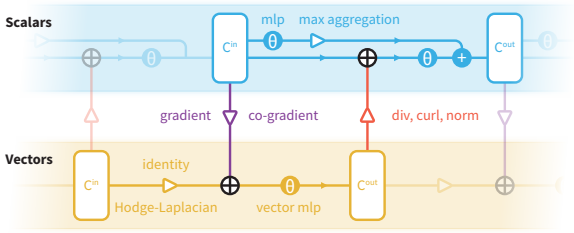


Figure 3.4: Schematic of DeltaConv.

the vector MLP is applied to $\mathbf{V}$ as follows

$$\mathbf{V}' = \sigma(\mathbf{V}\mathbf{W}), \tag{3.7}$$

where $\mathbf{W} \in \mathbb{R}^{C^{(l)} \times C^{(l+1)}}$ is a weight matrix and σ is a non-linearity applied to vector norms. Matrix multiplication with $\mathbf{W}$ linearly combines the vector features but the individual coefficients of a vector are not mixed. Before the vector MLP is applied, we concatenate the 90-degree rotated vectors to the input features. This allows the MLP to also rotate vector features and enriches the set of operators. For example, the 90-degree rotated gradient is the co-gradient. The vector MLP can learn to combine information from local neighborhoods (through the gradient and Hodge–Laplacian), as well as information from different channels (through the identity). A schematic overview of Equation 3.6 can be found in Figure 3.4.

While Equation 3.6 formulates DeltaConv in terms of MLPs and feature concatenation, an alternative perspective is to consider the operations in Equation 3.6 as linearly combining the elementary operators and composing them with non-linearities in-between to form anisotropic geometric operators.

3.3.7 Properties of DeltaConv

The building blocks of DeltaConv, such as the gradient, divergence, curl, and the combination with non-linearities allow DeltaConv to build nonlinear anisotropic convolution filters. This is illustrated by the example of the Perona–Malik filter in Figure 3.2. The vector stream also allows DeltaConv to process vector features and their relative directions directly with the appropriate operators.

DeltaConv is formulated in terms of smooth differential operators and is not restricted to a specific surface representation. In this work, we implement DeltaConv for point clouds and images. However, the concepts generalize to other representations. For example, an implementation for meshes could be done using finite element discretizations [20] or discrete exterior calculus [35].

DeltaConv is coordinate-independent, meaning that the weights used in DeltaConv do not depend on the choices of tangent bases. For

example, a forward pass on a shape with one choice of bases leads to the same output and weight updates when run with different bases. The coordinate-independence follows from the fact that all elementary operations in DeltaConv, such as applying geometric operators and vector MLPs, are coordinate-independent. It is known from differential geometry that one obtains the same results with geometric operators, no matter which basis is chosen [175]. This property is preserved by the discretization of the operators and thus inherited by DeltaConv.

Finally, each of the building blocks of DeltaConv is isometry invariant. That means DeltaConv does not change if a shape is isometrically deformed. This property can be beneficial for tasks where shapes are rigidly or non-rigidly deformed. If the surface orientation is flipped, rotations in the tangent plane are flipped as well. DeltaConv is robust to this if only the gradient and divergence are used.

3.4 Experiments

We validate our approach with comparisons to state-of-the-art approaches on classification and segmentation. In addition, we perform ablation studies to provide more insight into the effect of the vector stream on anisotropy, accuracy, and efficiency.

3.4.1 Implementation details

In our experiments we use network architectures based on DGCNN [236]. We replace each EdgeConv block with a DeltaConv block (Figure 3.4) and do not use the dynamic graph component. Thus, the networks operate at a single scale on local neighborhoods. Despite this simple architecture, DeltaConv achieves state-of-the-art results. To show what architectural optimizations mean for DeltaConv, we also test the U-ResNet architecture used in KPFCNN [224] but with the convolution blocks in the encoder replaced by DeltaConv blocks. In the downsampling blocks used by these networks, we pool vector features by averaging them with parallel transport [244]. More details are provided in the supplemental material. Code is available at `https://github.com/rubenwiersma/deltaconv`.

Data transforms. A k-nn graph is computed for every shape. This graph is used for maximum aggregation in the scalar stream. It is reused to estimate normals when necessary and to construct the gradient. For each experiment, we use xyz-coordinates as input to the network and augment them with a random scale and translation, similar to previous works. Some datasets require specific augmentations, which are detailed in their respective sections.

Training. The parameters in the networks are optimized with stochastic gradient descent (SGD) with an initial learning rate of 0.1,

Table 3.1: Classification results on ModelNet40.

Method	Mean Class Accuracy	Overall Accuracy
PointNet++ [187]	-	90.7
PointCNN [133]	88.1	92.2
DGCNN [236]	90.2	92.9
KPConv deform [224]	-	92.7
KPConv rigid [224]	-	92.9
DensePoint [144]	-	93.2
RS-CNN [145]	-	93.6
GBNet [189]	91.0	**93.8**
PointTransformer [271]	90.6	93.7
PAConv [255]	-	93.6
Simpleview [75]	-	93.6
Point Voxel Transformer [264]	-	93.6
CurveNet [254]	-	**93.8**
DeltaNet (ours)	**91.2**	**93.8**

momentum of 0.9 and weight decay of 0.0001. The learning rate is updated using a cosine annealing scheduler [147], which decreases the learning rate to 0.001.

3.4.2 Classification

For classification, we study ModelNet40 [253], ScanObjectNN [227], and SHREC11 [135]. With these experiments, we aim to demonstrate that our networks can achieve state-of-the-art performance on a wide range of challenges: point clouds sampled from CAD models, real-world scans, and non-rigid, deformable objects.

ModelNet40 The ModelNet40 dataset [253] consists of 12,311 CAD models from 40 categories. 9,843 models are used for training and 2,468 models for testing. Each point cloud consists of 1,024 points sampled from the surface using a uniform sampling of 8,192 points from mesh faces and subsequent furthest point sampling (FPS). We use 20 neighbors for maximum aggregation and to construct the gradient and divergence. Ground-truth normals are used to define tangent spaces for these operators and the regularizer is set to $\lambda = 0.01$. As input to the network, we use the xyz-coordinates. The classification architecture is optimized for 250 epochs. We do not use any voting procedure and list results without voting.

The results for this experiment can be found in Table 3.1. DeltaConv improves significantly on the most related maximum aggregation operators and is on par with or better than state-of-the-art approaches.

ScanObjectNN ScanObjectNN [227] contains 2,902 unique object instances with 15 object categories sampled from SceneNN [96] and ScanNet [37]. The dataset is enriched to $\sim 15,000$ objects by preserving or removing background points and by perturbing

Method	NO BG	BG	T25	T25R	T50R	T50RS
3DmFV [10]	73.8	68.2	67.1	67.4	63.5	63.0
PointNet [186]	79.2	73.3	73.5	72.7	68.2	68.2
SpiderCNN [258]	79.5	77.1	78.1	77.7	73.8	73.7
PointNet++ [187]	84.3	82.3	82.7	81.4	79.1	77.9
DGCNN [236]	86.2	82.8	83.3	81.5	80.0	78.1
PointCNN [133]	85.5	86.1	83.6	82.5	78.5	78.5
BGA-PN++ [227]	-	-	-	-	-	80.2
BGA-DGCNN [227]	-	-	-	-	-	79.9
GBNet [189]	-	-	-	-	-	80.5
GDANet [256]	88.5	87.0	-	-	-	-
DRNet [188]	-	-	-	-	-	80.3
DeltaNet (ours)	**89.5**	**89.3**	**89.4**	**87.0**	**85.1**	**84.7**

Table 3.2: Classification results on ScanObjectNN.

bounding boxes. The variant without background points is tested without any perturbations (NO BG). The variant with background points is both tested without (BG) and with perturbations: Bounding boxes are translated (T), rotated (R), and scaled (S) before each shape is extracted. This means that some shapes are cut off, rotated, or scaled. T25 and T50 denote a translation by 25% and 50% of the bounding box size, respectively.

We use a modified version of the classification architecture with four convolution blocks with the following output dimensions: 64, 64, 64, 128. This setup matches the architecture used for DGCNN in [227]. Normals are estimated with 10 neighbors per point and the operators are constructed with 20 neighbors and $\lambda = 0.001$. As input, we provide the xyz-positions, augmented with a random rotation around the up-axis and a random scale $S \in \mathcal{U}(4/5, 5/4)$. The network is trained for 250 epochs.

Our results are compared to those reported by the authors of ScanObjectNN (row 1-8) [227] and other recent approaches in Table 3.2. We find that our approach outperforms all networks for every type of perturbation, including networks that explicitly account for background points.

SHREC11 The SHREC11 dataset [135] consists of 900 non-rigidly deformed shapes, 30 each from 30 shape classes. This experiment aims to validate the claim that our approach is well suited for non-rigid deformations. Like previous works [85, 244, 201], we train on 10 randomly selected shapes from each class and report the average over 10 runs. We sample 2048 points from the simplified meshes used in MeshCNNs experiments [85] and use 20 neighbors and mesh normals to construct the operators ($\lambda = 0.001$). As input, we provide xyz-coordinates, which are randomly rotated along each axis. We decrease the number of parameters in each convolution of the classification architecture to 32, since the dataset is much smaller than other datasets. The network is trained for 100 epochs. We find

Table 3.3: Classification results on SHREC11.

Method	Accuracy
MeshCNN [85]	91.0
HSN [244]	96.1
MeshWalker [121]	97.1
PD-MeshNet [160]	99.1
HodgeNet [212]	94.7
FC [162]	99.2
DiffusionNet (xyz) [201]	99.4
DiffusionNet (hks) [201]	99.5
DeltaNet (ours)	**99.6**

that our architecture is able to improve on state-of-the-art results (Table 3.3), validating the effectiveness of our intrinsic approach on deformable shapes.

Table 3.4: Part segmentation results on ShapeNet.

Method	Mean inst. mIoU
PointNet++ [187]	85.1
PointCNN [133]	86.1
DGCNN [236]	85.2
KPConv deform [224]	86.4
KPConv rigid [224]	86.2
GDANet [256]	86.5
PointTransformer [271]	86.6
PointVoxelTransformer [264]	86.5
CurveNet [254]	86.8
DeltaNet (ours)	86.6
Delta-U-ResNet (ours)	**86.9**

3.4.3 Segmentation

For segmentation, we evaluate our architecture on ShapeNet (part segmentation) [263]. ShapeNet consists of 16,881 shapes from 16 categories. Each shape is annotated with up to six parts, totaling 50 parts. We use the point sampling of 2,048 points provided by the authors of PointNet [186] and the train/validation/test split follows [26]. The operators are constructed with 30 neighbors and ground-truth normals to define tangent spaces ($\lambda = 0.001$). The xyz-coordinates are provided as input to the network, which is trained for 200 epochs. During testing, we evaluate each shape with ten random augmentations and aggregate the results with a voting procedure. Such a voting approach is used in the most recent works that we compare with.

The results are shown in Table 3.4, where our approach, especially the U-ResNet variant, improves upon the state-of-the-art approaches on the mean instance IoU metric and in many of the shape categories (full breakdown in the supplemental material). For each category, DeltaConv is either comparable to or better than other architectures and significantly better than the most related intrinsic approaches (PointNet++ and DGCNN). In Figure 3.5, we provide feature visualizations to give an idea of the features derived by the network.

3.4.4 Ablation Studies

We aim to validate the claim of anisotropy, isolate the effect of the vector stream, validate the choices to regularize and normalize the gradient and divergence operators, and investigate the impact of our approach on the timing and parameter counts of these networks.

Anisotropy To validate that DeltaConv supports anisotropy, we train a network to mimic anisotropic diffusion [181]. A ResNet [87] with 16 layers and 16 channels in the hidden layers is trained for 100 iterations with Adam [111] to match a target image generated with 20 anisotropic diffusion steps. In each diffusion step, the gradients are scaled with $\exp(-(|v|/0.05)^2)$. We vary the convolution blocks in the network with the ones from DiffusionNet [201], EdgeConv [236], PointNet++ [187], GCN [112], and regular image CNNs. For DiffusionNet, we set the diffusion time to a fixed value, as we are interested in the ability of the convolution to derive anisotropic filters through its gradient features. For all other convolutions, the

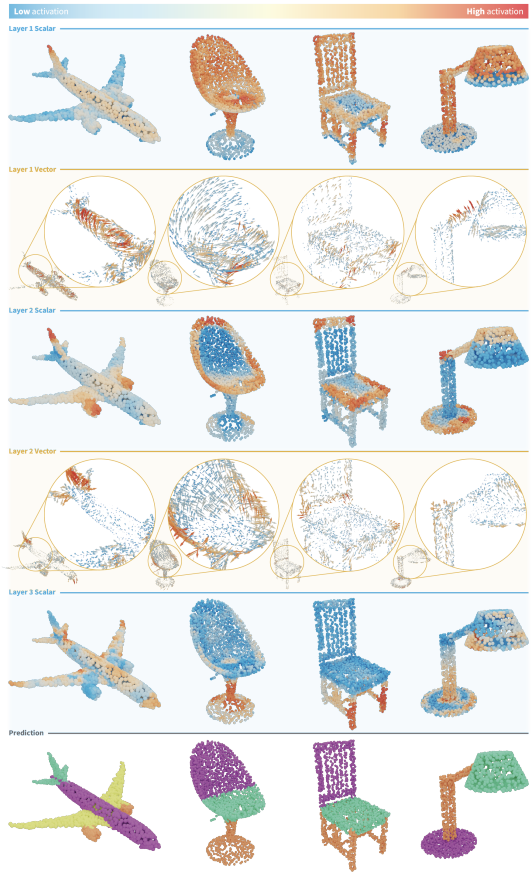


Figure 3.5: For each layer of the network, we show how a single scalar- or vector-feature varies over shapes in ShapeNet. The last row shows the output of the network. The features tend to activate on similar regions.

neighborhoods are 3x3 pixel blocks. The results are shown in Figure 3.2 and in the supplement. DeltaConv achieves a good match. The other operators tend to blur the image or produce artifacts. For PointNet and EdgeConv, this is likely due to the variable nature and sharpness of the maximum aggregation. DiffusionNet lacks the divergence and curl operators and does not maintain a vector stream, which is necessary to analyze the relative directions of vector features in local neighborhoods.

Effectiveness of vector stream To study the benefit of the vector stream and its effect on different types of intrinsic scalar convolutions, we set up three different scalar streams: (1) a Laplace–Beltrami operator, $\Delta = -\text{div}\,\text{grad}$, (2) GCN [112], and (3) maximum aggregation (Equation 3.1). We test three variants of each network: (1) only scalar stream, (2) scalar stream with the number of parameters adjusted to match a two-stream architecture, and (3) both the scalar and vector stream.

Table 3.5: Ablations of DeltaConv on ShapeNet (Seg) and ModelNet40 (M40) with varying scalar streams.

Scalar Convolution	Vector Stream	Match # params	Seg mIoU	M40 mcA	M40 OA
Laplace–Beltrami	-	-	82.5	86.1	90.4
	-	✓	82.5	87.1	90.6
	✓	-	**84.9**	**89.4**	**92.2**
GCN	-	-	81.1	87.3	90.4
	-	✓	81.2	87.3	90.8
	✓	-	**85.1**	**90.6**	**92.8**
Max aggregation	-	-	85.7	89.2	92.2
	-	✓	85.7	89.5	92.6
	✓	-	**86.1**	**91.2**	**93.8**

Table 3.6: Timing and parameter counts for classification on ModelNet40. The timing for training and inference includes all necessary precomputations.

Convolution	Data Transform	Training	Backward	Inference	# Params
DeltaConv (Lapl.)	k-nn + ops	80ms	5ms	80ms	2,036,938
DeltaConv	k-nn + ops	130ms	60ms	125ms	2,037,962
EdgeConv	k-nn	196ms	147ms	186ms	1,801,610

We test each configuration on ModelNet40 and ShapeNet. For both of these tasks, we use the DGCNN base architecture. The model for ShapeNet is trained for 50 epochs to save on training time and no voting is used, which results in slightly lower results than listed in Table 3.4. The results are listed in Table 3.5. We find that the vector stream improves the network for each scalar stream for both tasks, reducing the error between $19-25\%$ for classification and $3-21\%$ for segmentation. For maximum aggregation on ShapeNet, the improvements are lower, but still considerable, given the rate of progress on this dataset over the last few years. Simply increasing the number of parameters in the scalar stream does not yield the same improvement as adding the vector stream, showing that the vector-valued features are of meaningful benefit. Maximum aggregation in the scalar stream yields the highest accuracy.

Timing and parameters In our method section we argue that computing the gradient matrix is lightweight and that the simplified maximum aggregation operator is significantly faster than edge-based operators in PointNet++ and DGCNN. The main bottleneck in these convolutions is maximum aggregation over each edge. In this experiment, we demonstrate this by reporting the time it takes to train and test the classification network on one batch of 32 shapes with 1,024 points each. This includes all precomputation steps, such as computing the k-nearest neighbor graph ($\sim$ 15ms) and constructing the gradient and divergence operators ($\sim$ 30ms). The EdgeConv network is tested without a dynamic graph component, so that only the effect of precomputation and convolutions remains. All timings are obtained on the same machine with an NVIDIA RTX 2080Ti after a warm-up of 10 iterations. We implemented each

method in PyTorch [179] and PyTorch Geometric [63]. The results are listed in Table 3.6. We find that our network only increases the number of parameters by 10%. Our network is significantly faster than the edge-based convolution: 1.5× faster in training and inference and 2.5× faster in the backward pass. DeltaConv with a Laplacian in the scalar stream is even faster: > 2× faster in training and inference and 30× faster in the backward pass.

Gradient regularization and normalization In our method section, we argue that the least-squares fit for constructing the gradient and divergence should be regularized and the operators should be normalized. In this experiment, we intend to validate these choices. We train a model that is entirely based on our gradient operator, with a Laplace–Beltrami operator in the scalar stream. This means that every spatial operator in the network is influenced by regularization and scaling. The model is trained on the ModelNet40 for 50 epochs. The results are listed in Table 3.7. We notice a considerable difference between our approach with- and without regularization. There is a 2.8 percentage point decrease in mean class accuracy and 1.7 percentage point decrease in overall accuracy when the operator is not normalized.

Table 3.7: Classification accuracy on ModelNet40 with and without regularization and normalization.

λ	Norm.	Mean Class Acc.	Overall Acc.
10^{-32}	✓	85.2	90.3
10^{-2}	-	86.6	90.5
10^{-2}	✓	**89.4**	**92.2**

3.5 Conclusion

In this work, we propose DeltaConv, a new convolutional layer for point cloud CNNs that is capable of extracting and processing directional features. DeltaConv separates features into a scalar- and vector stream and uses linear combinations and compositions of a selected set of geometric operators from vector calculus to map between and along the streams. This construction allows DeltaConv networks to learn anisotropic convolutions fitting to the data and task at hand. We demonstrate improved performance on a wide range of tasks, showing the potential of using DeltaConv in a learning setting on point clouds. We hope that this work will provide insight into the functionality and operation of neural networks for point clouds and spark more work that combines learning approaches with powerful tools from geometry processing.

We limit our study to analysis tasks. While we do not think it is impossible to adapt our operators for generative tasks, it is unclear if and when the operators should be recomputed when a surface is generated. Our work opens up interesting possibilities for future work. Besides exploring more applications of the vector stream, we want to test our approach on other surface discretizations and other manifolds (e.g., hyperbolic spaces and higher dimensional spaces) for which these operators are available, and also intend to study how other variants of the scalar stream impact the network.

4 A Fast Geometric Multigrid Method for Curved Surfaces

We introduce a geometric multigrid method for solving linear systems arising from variational problems on surfaces in geometry processing, Gravo MG. Our scheme uses point clouds as a reduced representation of the levels of the multigrid hierarchy to achieve a fast hierarchy construction and to extend the applicability of the method from triangle meshes to other surface representations like point clouds, nonmanifold meshes, and polygonal meshes. To build the prolongation operators, we associate each point of the hierarchy to a triangle constructed from points in the next coarser level. We obtain well-shaped candidate triangles by computing graph Voronoi diagrams centered around the coarse points and determining neighboring Voronoi cells. Our selection of triangles ensures that the connections of each point to points at adjacent coarser and finer levels are balanced in the tangential directions. As a result, we obtain sparse prolongation matrices with three entries per row and fast convergence of the solver. Code is available at* `https://graphics.tudelft.nl/gravo_mg`.

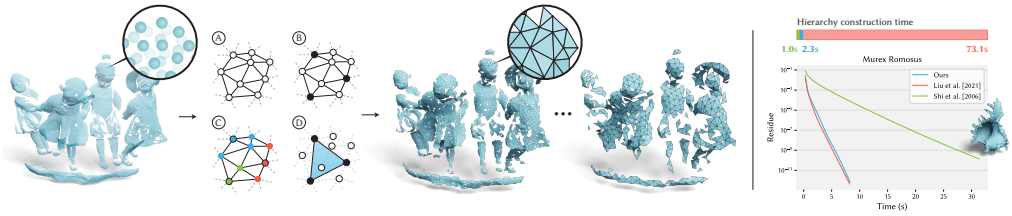


Figure 4.1: Illustration of our hierarchy construction for one level. Starting from a mesh or point cloud, A) we construct a neighbor graph on the surface. B) We then sample a spatially uniform set of nodes, C) compute the graph Voronoi diagram of the samples, and D) project unsampled points onto triangles formed by edges between Voronoi neighbors. This is repeated for every level. Right: Comparison of run time for solving a Laplace system on a triangle mesh. Our hierarchy construction is fast, while achieving similar solver performance to the state-of-the-art.

* This chapter is based on the paper "A Fast Geometric Multigrid Method for Curved Surfaces" published in SIGGRAPH 2023 Conference Proceedings [245].

4.1 Introduction

Many geometry processing methods are based on variational problems and partial differential equations on curved surfaces. The discretization of these problems leads to sparse linear systems to be solved. One class of efficient solvers are Geometric Multigrid (GMG) methods, which use iterative solvers on a hierarchy of grids. They are more efficient than alternatives, such as sparse direct solvers, in many application scenarios [140]. While geometric multigrid solvers are well-studied for regular grids in Euclidean domains, the construction of effective geometric multigrid hierarchies remains challenging for irregular meshes on curved domains.

We distinguish two approaches to the design of GMG methods on curved surfaces. The first approach is to construct a hierarchy of meshes by mesh coarsening and then mapping between the meshes. This approach obtains efficient prolongation operators that lead to fast convergence. A recent example is the intrinsic multigrid scheme by Liu et al. [140]. The downside of this approach is a costly hierarchy construction. The second approach is to represent levels by graphs constructed by coarsening the edge graph of the input mesh [207]. This approach results in a fast construction but slower convergence.

We propose a new GMG method combining the strengths of both approaches. On the one hand, we use point clouds and neighbor graphs to represent levels, enabling a fast hierarchy construction. On the other hand, we use geometric operations to create local triangulations when constructing the prolongation operators for fast convergence. Our method solves linear systems as fast as the scheme of Liu et al., while reducing hierarchy-construction time by more than an order of magnitude. Moreover, our method is more generally applicable as it can be used not only for manifold triangular meshes but also for other discrete surface representations such as point clouds, non-manifold meshes, and polygonal meshes. Thus, we can solve systems set up with discrete differential operators for these representations, which were developed in recent years [136, 202, 6]. Our hierarchy construction is more expensive compared to Shi et al. [207]. Yet, the solving time is most often reduced more than the increase in hierarchy construction. This benefit increases for applications where multiple systems need to be solved.

The technical novelty of our method lies in a geometric multigrid method that is point-based, while still incorporating the geometry of the underlying surface. Our guiding idea is to construct intrinsic Delaunay triangulations on points sampled from the surface. Every other point can then be mapped from- and to the sampled points using barycentric coordinates in the intrinsic triangles. To get a fast and practical approach, we transfer this idea to a point-cloud setting. For every level in the hierarchy, we start by sampling points from

the previous level using a fast uniform sampling strategy. Next, we compute graph Voronoi diagrams on the finer level using the sampled points as seeds and construct a neighborhood graph based on Voronoi cell adjacencies. Mimicking Delaunay triangulations, we construct triangles from the edges of the Voronoi adjacency graph. Each point of the finer level is projected to its closest triangle and barycentric coordinates are used for prolongation. This construction leads to sparse prolongation matrices with at most three entries per row and hence to fast prolongations and restrictions. The use of graph Voronoi cells ensures that the prolongation matrix and its transpose (the restriction matrix) contain entries corresponding to neighbors that are well-distributed over the tangential directions. We name our hierarchy construction Gravo MG, for graph Voronoi multigrid.

We evaluate Gravo MG in ablations and comparisons to [140], [207], and algebraic multigrid methods. Furthermore, we demonstrate the benefits of our scheme over sparse direct solvers in application scenarios.

4.2 Related Work

Geometric multigrid Multigrid methods [18] are among the most efficient iterative methods for solving linear systems. We call them *geometric* multigrid methods if the hierarchy construction is exclusively on the domain and no information is used about the system to be solved. GMG methods on regular grids are well studied [83] and used in graphics, *e.g.*, for fluid simulation [156, 47], image processing [180, 119], and surface reconstruction [109, 107]. GMG methods for irregular grids on Euclidean domains are used for the simulation of cloth (2D) [104, 237] and elastic objects (3D) [71, 176].

In this work, we consider GMG methods for curved surfaces. Since the domain is no longer a Euclidean space but a curved manifold, methods from the Euclidean setting do not transfer directly and new methods are needed. Existing GMG methods on surfaces focus on triangle mesh representations of discrete surfaces. If the mesh is already equipped with a hierarchy, for example from a subdivision method, it can be used directly for a multigrid method [77]. However, usually, only a fine-scale mesh is given and a hierarchy must be built. Based on earlier work on multiresolution representations of triangle meshes, [95, 114], edge collapses are used to create multigrid hierarchies in [192, 5, 168]. The prolongation operators are defined by weighted averaging with the one-ring neighbors. There are two approaches to guaranteeing that each vertex in a finer level has at least one neighbour in the coarser level: either the coarsening process is restricted to only collapsing edges, so that a maximal independent set of vertices (MIS) is removed [168, 5], or the edge collapses are restricted so that a MIS is preserved [5]. Liu et al. [140]

introduce an intrinsic multigrid scheme that uses edge coarsening to create the meshes for the different levels and maintains bijective mappings between the meshes on consecutive levels. The map between two meshes is used to define the prolongations. The map assigns to each vertex of the finer mesh a point in a triangle of the coarser mesh and linear interpolation in the triangle is used for prolongation. The resulting prolongation matrix has at most three entries per row. An alternative to mesh coarsening is to use graph coarsening for hierarchy construction [207, 208]. A multigrid scheme for the computation of Laplace–Beltrami eigenpairs on surfaces is introduced in [166]. The hierarchy used for the eigenproblem, however, is much coarser than the hierarchies used for solving linear systems: only two or three levels are used. A multigrid solver for the computation of harmonic foliations on surfaces is introduced in [235].

Algebraic multigrid Algebraic multigrid (AMG) methods [19, 217] are an alternative to GMG. They use the matrix of the linear system to be solved to build the hierarchy instead of using the domain. This has the advantage that AMG can be used for problems coming from arbitrary domains. Nevertheless, AMG methods need to rebuild the hierarchy when the system matrix changes, whereas GMG methods only need to rebuild the hierarchy when the domain changes. An efficient multigrid preconditioner specifically for Laplace systems on images and meshes was introduced in [118]. Although fast, it has the disadvantage of requiring the Laplace matrices to have only non-positive off-diagonal entries, which is often not satisfied by mesh Laplacians, such as the cotangent-Laplacian [183].

Direct solvers Sparse direct solvers [38] are reliable, accurate, and commonly used for Laplace systems in geometry processing. Once a factorization of a matrix is computed, these solvers can solve multiple systems with the same matrix but different right-hand sides. In special cases, such as low-rank changes of the matrix, the factorization can be updated efficiently [29, 89, 91]. However, substantial changes require a new factorization. A disadvantage of these solvers is that they do not scale well neither in terms of memory requirements nor computation time. In Section 4.5, we compare the performance of our method to direct solvers in different scenarios.

4.3 Background: Multigrid Solver

Multigrid solvers use a hierarchy of grids to solve systems of equations. Iterative solvers converge at different speeds for different scales, depending on the resolution of the grid on which they operate. Thus, by performing iterations on different grids, a multigrid scheme

extends the range in which the solver converges particularly fast. Here, we describe a multigrid solver, which will later be used to evaluate our proposed hierarchy and prolongation operators.

We consider the multigrid solver in Algorithm 1. To solve an n-dimensional linear system $Ax = b$ for x, it operates on a multigrid hierarchy with λ levels, where level 1 is the finest and level λ is the coarsest level. A function on the l^{th} grid is represented by a vector in $\mathbb{R}^{n_l}$, where the grid has n_l degrees of freedom. The mappings between the grids are realized by prolongation and restriction matrices. The prolongation matrices $P_l \in \mathbb{R}^{n_l \times n_{l+1}}$ map from level $l+1$ to level l. We use the transposed matrices of the prolongation matrices $P_l^\top$ as restriction matrices. The advantage is that a symmetric matrix A implies that the linear systems in the coarse grid correction, which involve the restricted matrices $P_l^\top A_l P_l$, are also symmetric.

The multigrid solver first builds the prolongation matrices. We keep this step abstract at this point but discuss it in detail in the following section. In the next step, lines 3-6, the restricted matrices for all levels are constructed. After the precomputation, multigrid iterations are executed until convergence of the solution. The multigrid iterations traverse the hierarchy from fine to coarse and back. This process is called a V-cycle and is simple but effective. Alternatively, instead of directly going up to the coarsest grid, one could first go back to finer grids. Such strategies can help to counteract error accumulation when several levels are traversed and thereby reduce the required number of multigrid iterations. On the other hand, the V-cycle is fast. The multigrid iterations, Algorithm 2, apply relaxation steps before and after the coarse grid correction. We use Gauss–Seidel iterations for this. Alternatives are schemes such as Jacobi iterations or conjugate gradient iterations. The number of Gauss–Seidel iterations applied in the pre and post relaxations is specified by the parameters ν_{pre} and ν_{post}. For V-cycles, one sets $\nu_{pre} = \nu_{post}$.

The norm used for the convergence test in line 9 of Algorithm 2 depends on the context. A common choice is the standard norm of $\mathbb{R}^n$. For the Poisson and smoothing problems, we use a mass-weighted 2-norm [239].

Algorithm 1: Multigrid solver

Input: Matrix $A \in \mathbb{R}^{n \times n}$, initial vector $x \in \mathbb{R}^n$, right-hand side $b \in \mathbb{R}^n$, error tolerance ε, number of levels λ, numbers of pre/post-relaxations steps ν_{pre}, ν_{post}
Output: Solution $x \in \mathbb{R}^n$ to the linear system $Ax = b$

Function `Multigrid`$(A, x, b, \varepsilon, \lambda, \nu_{pre}, \nu_{post})$**:**
  Build prolongation matrices $P_1, P_2, P_3 \ldots, P_\lambda$
  $A_1 \leftarrow A$
  for $l \leftarrow 2$ *to* λ **do**
    $A_l \leftarrow (P_{l-1})^\top A_{l-1} P_{l-1}$ // Build level l matrix
  repeat
    $x \leftarrow$**MGI**$(x, b, 1)$
  until $\|Ax - b\| \leq \varepsilon \|b\|$ *// Convergence test*
  return x
End Function

Algorithm 2: Multigrid Iteration (V-cycle)

Input: Current iterate $x \in \mathbb{R}^{n_l}$, right-hand side $b \in \mathbb{R}^{n_l}$, level l
Output: New iterate $x \in \mathbb{R}^{n_l}$

Function `MGI`(x, b, l)**:**
  if $l < \lambda$ **then**
    $y \leftarrow$**Relax**(A_l, x, b, ν_{pre}) // Pre-relaxation
    $u \leftarrow$**MGI**$(0, P_l^\top (b - A_l y), l + 1)$ // Recursive call
    $x \leftarrow$**Relax**$(A_l, x + P_l u, b, \nu_{post})$ // Post-relaxation
  else
    Solve $A_\lambda x = b$ using a direct solver
  return x
End Function

4.4 Hierarchy Construction

Our goal is to design a hierarchy construction that is faster than the intrinsic multigrid method by Liu et al. [140]. It should be compatible with point clouds and general surface representations, while maintaining fast convergence during solving. Before giving an overview of our method, we revisit the idea that guided our design.

In the scheme of Liu et al., each level is represented by a mesh and mappings to adjacent levels. An intrinsic multigrid approach can alternatively represent levels by multiple *intrinsic* triangulations of the same surface. For example, each level can be a point sampling with corresponding intrinsic Delaunay triangulation. This idea, however, does not reflect a fast and more general construction. To achieve this, we transfer the idea into a point-cloud setting.

4.4.1 Overview

Our approach takes as input a set of point locations V sampled from a surface and a set of edges E between these points denoting local neighborhoods. For a mesh $\{V, E, F\}$, we use the vertices V

and edges E. For a point cloud, edges could be taken from a radius graph or the 1-ring in a local Delaunay triangulation [202].

The algorithm outputs a sequence of sparse prolongation matrices P_l, mapping signals from n_{l+1} points to n_l points, where $n_{l+1} < n_l$. Relating n_{l+1} and n_l, we refer to points in level n_{l+1} as *coarse points* and points in n_l as *fine points*. Level 1 are the input points.

Algorithm overview The hierarchy is constructed one level at a time. For each level, the algorithm takes the input graph from level l, $\{V_l, E_l\}$, and outputs the graph in level $l+1$, $\{V_{l+1}, E_{l+1}\}$. The algorithm also produces the prolongation matrix P_l. This is repeated until level λ is reached. Here, we describe one such step. Figure 4.1 provides a corresponding visual overview.

First, the point cloud is subsampled using a fast greedy algorithm that aims to enforce a minimum edge length in the next graph (subsection 4.4.2). Next, we create a graph Voronoi diagram, where the coarse points (sampled points) act as Voronoi centers and the fine points are the loci of the Voronoi cells. We then seek a mapping from the coarse points to the fine points. Mimicking the construction of a Delaunay triangulation as the dual of a Voronoi diagram, we construct a neighbor relation of the graph Voronoi cells (subsection 4.4.3) and compute all the triangles formed by the edges between Voronoi cell centers (subsection 4.4.4). Finally, the fine points are projected onto these triangles to find the triangle closest to the fine point. The neighbor relations of the graph Voronoi diagram are then used as edges for the next level E_{l+1}.

4.4.2 Sampling

Every level contains fewer points than the previous and the sampling should be spatially uniform [140, 207]. We seek a dense sample set S, in which no pair of points is closer than a fixed distance r. To find such a set, we use an algorithm based on the maximum independent set: we sweep once over V, keeping track if points are eligible for addition to S. Initially, all points are eligible. If a point p is eligible, we add it to S and mark the points within geodesic distance r of p as ineligible. The radius r is based on the fraction $\phi < 1$ of points we wish to keep and the average edge length $\hat{e}$

$$r = \phi^{-\frac{1}{3}} \hat{e}. \tag{4.1}$$

In our experiments, $\phi = 1/8$ and we stop at 1000 points, yielding roughly $\log_8(N/1000)$ levels. We only search for nearby points in the 2-ring, striking a good balance between construction speed and sampling quality. In paragraph 4.5.3 we validate that we reach the desired fraction of samples and Figure 4.2 demonstrates the uniformity of the resulting sampling and corresponding triangles.

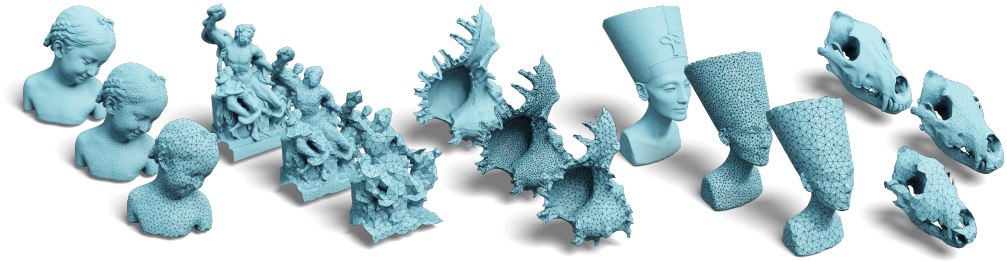


Figure 4.2: The input shapes and triangles in the last levels. Shapes: Bimba, Lakoon (non-manifold), Murex Romosus, Nefertiti, and Wolf Skull.

4.4.3 Neighbor graph

We use graph Voronoi diagrams [56] to define neighborhoods for the sampled points. Since we build the levels successively from fine to coarse, the neighbor graph $\{V_l, E_l\}$ is already built on level l when level $l+1$ is visited. The points V_{l+1} are the seeds of the graph Voronoi diagram in $\{V_l, E_l\}$. For each seed $i \in V_{l+1}$, the Voronoi cell consists of the points in V_l that are closer in graph distance to i than to all other points of V_{l+1}. The graph Voronoi diagram can be efficiently computed by a multisource Dijkstra algorithm. For points $i, j \in V_{l+1}$, we add an edge $\{i, j\}$ to E_{l+1}, if there is an edge in E_l that connects a point of the Voronoi cell of i with a point of the Voronoi cell of j.

4.4.4 Prolongation

Prolongation operators map functions on level $l+1$ to functions on level l, by matrices $P_l \in \mathbb{R}^{n_l \times n_{l+1}}$. The restrictions, mapping from level l to level $l+1$, are given as the transpose matrices $P_l^\top$. Important for the design of the prolongation matrices is their sparsity. The sparser the prolongation matrices, the sparser the restricted matrices $P_l^\top A P_l$, and the faster the mappings between the levels. To construct the prolongation, we use linear interpolation in triangles. Hereby, we get very sparse prolongations matrices. Other interpolation methods, such as radial basis functions or spline interpolations, would result in much denser prolongation matrices.

First, a set of candidate triangles on the coarse points is constructed. Every coarse point has a Voronoi cell on the finer level. Two coarse points i, j are connected by an edge $\{i, j\} \in E_{l+1}$ in the coarser level if their corresponding Voronoi cells are neighbors. We consider all triangles that can be constructed from these edges: all triplets $\{i, j, k\}$ such that $\{i, j\}, \{j, k\}, \{k, i\} \in E_{l+1}$.

The motivation to use these edges is the duality between (intrinsic) Voronoi diagrams and Delaunay triangulations (two points in a Delaunay triangulation are connected by an edge iff their Voronoi cells are adjacent). Since graph Voronoi cells are not continuous, but

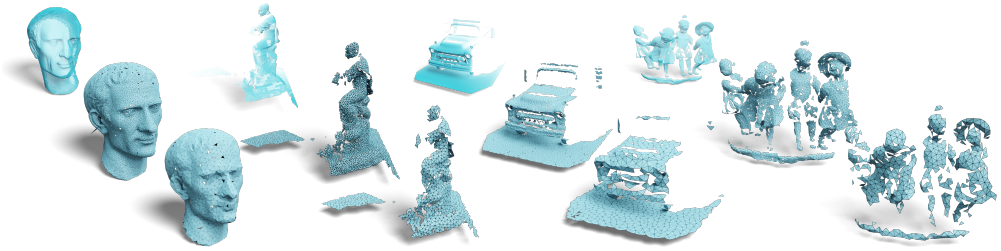


Figure 4.3: Input point clouds and considered triangles for the last two levels of the hierarchy. From left to right: Caesar, Ignatius, Truck, and Dancing Children.

approximations computed from a sampling, the triangles we obtain are not necessarily Delaunay triangles, and they do not necessarily form a manifold. However, as illustrated in Figure 4.2 and 4.3, we mostly get well-shaped triangles and a good coverage of the surface, even for point clouds.

To get the prolongation weights for a fine point p, we search for the closest candidate triangle. For efficiency, we restrict this search to the triangles that include the coarse point closest to p. The weights are the barycentric coordinates of the closest point to p in the selected triangle (this can be on an edge or a vertex). The barycentric coordinates of the projected point are then entered into the prolongation matrix.

Edge-cases In some cases, a suitable triangle cannot be found within the Voronoi neighborhood of the closest point. This might happen, for example, if all points in the neighborhood are (nearly) co-linear, or if the fine point falls outside of the triangles formed in the neighborhood. In these cases, we resort to finding the closest three points within the neighborhood and use inverse-distance weights. This is preferable over projecting to a single vertex, as it helps the spread of information during prolongation. In practice, this only happens in a fraction of cases (roughly 0.25%).

Reducing single-entry rows The resulting prolongation matrices are very sparse with maximally three non-zero entries per row. Since the coarse points are created by subsampling the fine points, the fine points that are sampled transfer their function value directly to the corresponding coarse point during prolongation. Therefore, there is only one entry in the corresponding rows. We obtain prolongation matrices with fewer single-entry rows by moving each sampled point to the mean of the points that form its graph Voronoi cell before we compute the closest point projections. In our experiments, we obtained a slight improvement of solving times with this strategy over not moving the coarse points.

4.5 Experiments

We evaluate Gravo MG and compare to state-of-the-art GMG methods and AMG approaches. For reference, we provide results for direct solvers. We also provide insights into design choices via ablation studies.

4.5.1 Implementation

Our multigrid-solver implementation builds on Liu et al.'s code, where the prolongation matrix definition is exchanged. In every experiment, we set the number of pre/post-relaxation steps $v_{pre} = v_{post} = 2$. The hierarchy construction uses custom routines built around Eigen [78] and only requires a matrix of points and an array of edges. The code for our solver is available as a C++ library and Python package, along with scripts to replicate the main tables and figures in this chapter: `https://graphics.tudelft.nl/gravo_mg`.

We use an Intel®Core™i9-9900 CPU (3.10GHz, 32GB memory). The code does not employ multithreading but could be parallelized. None of the methods in our comparisons are parallelized, except Pardiso. A discussion on the potential for parallelization of the solver we used can be found in Section 7 of Liu et al. [140].

4.5.2 Problems

In our ablations and comparisons, we test our approach on two standard problems that can be written as linear systems: data smoothing and Poisson problems. For meshes, both problems involve the cotan Laplace matrix S and the lumped mass matrix M, see [239]. In the case of point clouds and non-manifold meshes, we use the robust Laplacian by Sharp and Crane [202]. Data can be smoothed by solving

$$(M + \alpha S)x = My, \tag{4.2}$$

where y is the noisy input function and α a parameter that determines how much the data is smoothed. The Poisson problem is

$$(S + \eta M)x = My, \tag{4.3}$$

where y is a random vector. The term ηM is added to obtain a positive-definite system matrix. The parameter η is chosen to be very small, for example $\eta = 1 \times 10^{-6}$. Our solver terminates when tolerance ε is reached (line 9 of algorithm 1) or after a maximum number of iterations.

4.5.3 Ablation Studies

We would like to understand the effects of our design choices on the hierarchy construction and subsequent solving steps. We structure

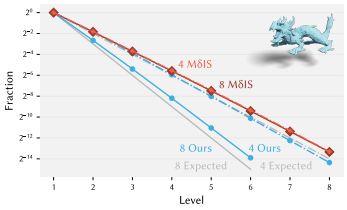


Figure 4.4: Comparison of the decay rate between MδIS used by Shi et al. [207] and our approach on XYZ dragon. The y-axis is in $\log_2$ scale.

these experiments along three themes: sampling, prolongation selection, and weighting. In each ablation, we compare variants of our approach on a fixed set of meshes and point clouds and run a data smoothing problem as detailed above. We smooth a random function with $\alpha = 1 \times 10^{-3}$ and tolerance 1×10^{-4}. Each variant is then evaluated in terms of time to construct the hierarchy, the number of iterations required to reach the target tolerance, and the total time.

Sampling We seek a sampling method that balances run-time during hierarchy construction and sampling quality. To validate that our approach effectively balances these demands, we compared our approach to random sampling, Poisson-disk-sampling (PDS), geodesic farthest point sampling (FPS), and maximal independent set (MIS) selection. For every method, we set the target ratio between levels to 1/8th. In Table 4.1, we observe that our approach is faster than the others when solving. When considering the full hierarchy construction, our approach is faster than every other sampling approach, because we perform part of the graph Voronoi diagram construction during sampling and have fewer points per level to consider than MIS.

We also compare decay rate between the maximal δ-independent set, used in [207], and our sampling. In Figure 4.4, we see that it is possible to decay much faster with our approach than MδIS, because it must always be a superset of the MIS. This is an advantage of our approach; we can perform faster and fewer iterations, while having a fast sampling time.

Prolongation selection Our approach uses triangles of coarse points for the prolongation operator. This results in sparse prolongation matrices that spread information in each tangential direction. In this ablation, we seek to support this choice. In Table 4.2, we compare our approach to the following variants that do not explicitly work with triangles: simply prolonging to the closest two, three, or four points from the graph Voronoi neighbors and picking three random points. We also test a variant that considers triangles without restricting to Voronoi edges, 'closest tri'. This results in less consistent triangulations, as shown in Figure 4.5. For each of the non-triangle selection approaches, we use inverse distance weights.

Table 4.1: Data smoothing timings with variations of the sampling step (*Smp*). We compare our approach to *random* sampling, Poisson Disk Sampling (*PDS*), geodesic Farthest Point Sampling (*FPS*), and Maximum-Independent Set (*MIS*). All timings are in seconds, unless otherwise specified.

		Ours		Random		PDS		FPS		MIS	
Model	#V	Smp	Solve	Smp	Solve	Smp	Solve	Smp	Solve	Smp	Solve
Brd Man	691k	0.08	**0.62**	0.01	1.01	0.47	0.63	1m	0.79	0.02	0.87
Rd Circ. Box	701k	0.08	**0.81**	0.01	1.30	0.39	1.10	1m	1.38	0.02	1.01
Nefertiti	1m	0.12	**1.10**	0.01	2.05	1.02	1.29	2m	1.65	0.04	1.15
Murex	1.8m	0.42	**3.02**	0.03	4.84	1.47	3.41	6m	4.83	0.11	3.68
XYZ Dragon	3.6m	0.35	5.72	0.06	12.95	2.48	**4.92**	27m	-	0.11	7.38

Table 4.2: Timings for hierarchy construction and solving on data smoothing with variations of the entries in the prolongation operator. *Ours* only considers triangles formed by Voronoi edges. The other variants either pick *n closest* points, pick *n random* points or pick the three Voronoi neighbors that form the *closest triangle*. All timings are in seconds.

		2 points			3 points												4 points		
		Closest			**Ours**			Random			Closest vert			Closest tri			Closest		
Model	#Vert	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve
Beard Man	691k	0.46	9	0.91	0.64	4	**0.62**	0.42	26	2.91	0.50	5	0.72	0.87	5	0.77	0.51	5	0.83
Red Circular Box	701k	0.49	10	1.03	0.67	6	**0.79**	0.49	31	3.45	0.52	7	0.95	1.01	8	1.06	0.58	7	1.01
Nefertiti	1m	0.67	9	1.68	0.93	4	**1.05**	0.60	29	5.75	0.69	6	1.36	1.35	5	1.29	0.73	5	1.37
Murex Romosus	1.8m	1.80	11	4.15	2.64	5	**3.11**	1.70	40	15.07	2.07	6	3.60	3.82	6	3.41	1.97	6	3.91
XYZ Dragon	3.6m	2.41	15	7.55	3.46	9	**5.72**	2.28	45	25.10	2.61	14	8.35	4.45	16	8.76	2.68	11	7.32

We observe that our approach solves faster than all other variants, while increasing the hierarchy construction time only a bit.

Weighting We project the fine points onto the triangles formed by coarse points and use barycentric weights as predictors for the value of the fine point. Previous works suggest that the choice of weighting schemes has little effect on convergence times [5, 207], while Liu et al. argue that the weighting scheme is crucial for some shapes. In Table 4.3, we compare our approach with uniform weights and inverse distance weights alongside a variant where we do not shift the coarse points to the barycenters. We observe that our approach works best with barycentric coordinates (*Ours*). Inverse-distance weights are not far behind. Shifting coarse points has benefits for some, but not all shapes. This is not the core contribution of our work and could be left out in some cases. A benefit of not shifting coarse points is that each iteration is faster because the prolongation matrix contains more single-entry rows.

4.5.4 Comparisons

We compare our approach on a wide range of meshes and point clouds for a Poisson problem with $\eta = 1 \times 10^{-6}$ and target tolerance of 1×10^{-4}. The input function y is a random vector sampled from $\mathcal{N}(0, 1)$.

The shapes were selected to have at least 100k vertices and exhibit a wide variety: uniform meshes (*e.g.*, Nefertiti), non-uniform meshes (*e.g.*, Alfred Jacquemart, Indonesian statue), broken and non-manifold meshes. The meshes also exhibit detailed features (*e.g.*, XYZ dragon) and complex curvature (*e.g.*, Murex Romosus).

		Ours		Uniform		Inv. Dist.		No shift	
Model	#Vert	#It	Solve	#It	Solve	#It	Solve	#It	Solve
Beard Man	691k	4	0.63	12	1.30	5	0.71	4	**0.56**
Red Circular Box	701k	6	**0.80**	23	2.36	6	**0.80**	10	1.12
Nefertiti	1m	4	1.04	14	2.68	5	1.21	4	**0.97**
Murex Romosus	1.8m	5	3.09	16	6.54	6	3.37	5	**2.71**
XYZ Dragon	3.6m	9	**5.71**	23	12.23	9	5.74	18	9.28

Table 4.3: Timings for solving on data smoothing with variations of the weighting scheme. We compare barycentric coordinates (*Ours*) to *uniform* weights, *inv*erse *dist*ance weights, and barycentric coordinates without chaging the positions of the coarse points before projection (*No shift*). All timings are in seconds.

All the shapes are shown in Figure 4.6. We make no use of additional pre-processing steps, such as remeshing or fixing non-manifold edges: every mesh is used as-is in the highest resolution available from the respective sources. For the point clouds, we opted for high-resolution scanned data. The point clouds come from the Tanks and Temples benchmark dataset [113] and from range scans in the AIM@Shape repository [59].

Gravo MG is compared to the GMG solvers by Liu et al. and Shi et al., and the AMG methods Ruge–Stuben and Smoothed Aggregation. For reference, we list the timings of direct solvers. For Liu et al., we use their provided implementation. We reimplemented Shi et al. based on their paper. The latter mentions multiple weighting schemes, including uniform weights and inverse distance weights. We tested both and report the best-performing approach: inverse distance weights. For the AMG approaches, we use the implementation provided in PyAMG [9] with default settings provided by the package. We set the maximum number of iterations for all iterative solvers to 100, since more iterations would not change the overall picture regarding which solver is faster. The direct solver references are the Cholesky LLT factorization provided in Eigen and Intel®MKL's Pardiso solver, which is highly optimized and parallelized [100].

Our approach yields faster solving times for the majority of input meshes (Table 4.4). More results for manifold meshes are listed in the supplement in Table 1. On average, our construction is 36x faster than Liu et al. and only 1.8x slower than Shi et al.'s method. With regards to solving time, Liu et al. takes 3% more time on average for the Poisson problem and 7% for data smoothing and Shi et al. takes 274% more time for the Poisson problem and 81% for data smoothing. Note that we require less time for one iteration than Liu et al., because we use a higher decay rate (1/8 vs. 1/4). This is balanced out in most cases by a higher iteration count and the overall solving times are similar when we use a decay rate of 1/4.

GMG methods are most beneficial in settings where a user would iterate on the system matrix, but the benefit of using Gravo MG is already noticeable starting with the first solve. For all meshes larger

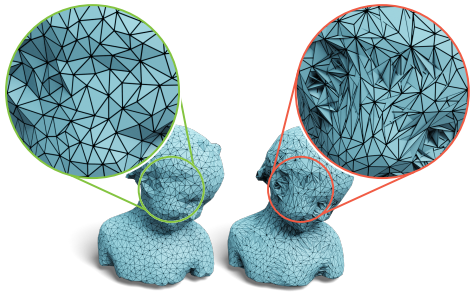


Figure 4.5: Using dual Voronoi triangles results in a more consistent set of candidate triangles than using all triangles in the coarse point's 1-ring.

Table 4.4: Comparison of our hierarchy construction and solver for a Poisson problem with $\eta = 1 \times 10^{-6}$ mass matrix coefficient and tolerance of 1×10^{-4}. Missing entries are not available for the given method. The maximum number of iterations for iterative solvers is set to 100.

		Gravo MG (Ours)			Liu et al.			Shi et al.			AMG-RS			AMG-SA			Eigen		Pardiso	
Model	#Vert	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Fact.	Subst.	Fact.	Subst.
MANIFOLD TRIANGULAR MESHES																				
Aim Dragon	152k	0.14	7	0.19	5.14	8	0.27	0.06	27	0.62	0.15	26	0.46	0.31	29	0.48	0.81	0.02	0.58	0.04
Blade Smooth	195k	0.15	4	0.17	5.87	3	0.17	0.09	18	0.60	0.18	100	2.42	0.40	42	0.94	0.76	0.02	0.69	0.04
Moses	258k	0.42	12	0.55	8.72	5	0.35	0.30	100	4.30	0.27	100	3.64	0.55	100	3.33	0.90	0.02	1.04	0.05
Julius Caesar	387k	0.30	11	0.58	12.10	17	1.09	0.16	28	1.54	0.39	100	4.89	0.77	70	2.79	5.07	0.06	1.29	0.09
Bimba	502k	0.43	7	0.65	15.58	6	0.74	0.24	48	4.16	0.51	100	6.97	1.15	69	4.45	3.54	0.05	2.13	0.10
Antique Head	651k	0.53	4	0.51	20.07	4	0.60	0.28	14	1.22	0.64	100	8.48	1.37	67	4.33	15.02	0.11	2.43	0.17
Beard Man	691k	0.59	4	0.56	22.15	3	0.58	0.26	14	1.43	0.52	100	7.07	1.46	14	0.96	24.57	0.14	2.72	0.19
Red Circular Box	701k	0.64	6	0.74	22.97	6	0.97	0.34	66	6.67	0.72	100	8.98	1.51	66	5.01	17.76	0.11	2.84	0.17
Dancing Children	724k	0.69	9	1.10	23.21	8	1.25	0.41	39	4.54	0.68	100	9.32	1.23	100	8.70	6.18	0.09	2.78	0.17
Ramses	826k	0.79	7	1.21	28.90	5	1.18	0.47	40	6.03	0.91	100	11.82	2.08	49	5.40	6.24	0.09	3.60	0.18
Nefertiti	1m	0.89	4	0.94	34.22	4	1.18	0.56	65	11.69	1.09	100	14.74	2.58	46	6.14	9.15	0.10	4.54	0.22
Isidore Horse	1.1m	1.12	11	1.90	35.60	5	1.27	0.50	70	11.03	1.11	100	14.60	2.50	88	10.72	24.01	0.17	4.56	0.28
Ram	1.3m	2.40	3	1.95	56.18	-	-	1.14	49	13.39	2.45	100	25.95	4.72	100	21.71	17.07	0.15	6.99	0.33
Murex Romosus	1.8m	2.32	6	2.85	73.05	5	3.32	0.99	63	20.79	2.38	100	29.83	5.08	58	15.28	40.06	0.26	9.13	0.44
XYZ Dragon	3.6m	3.24	9	5.32	121.97	7	5.32	1.57	55	28.95	3.16	100	43.75	9.14	75	30.54	77.62	0.69	15.88	0.94
NON-MANIFOLD TRIANGULAR MESHES																				
Lakoon	188k	0.16	8	0.29	-	-	-	0.09	41	1.33	0.21	100	2.77	0.47	48	1.12	0.42	0.01	0.71	0.04
Indonesian Statue	294k	0.26	11	0.58	-	-	-	0.16	64	3.15	0.30	100	3.92	0.63	100	3.55	0.92	0.03	1.18	0.06
Beethoven	383k	0.45	4	0.49	-	-	-	0.23	60	4.00	0.51	20	1.29	0.96	100	5.30	2.39	0.04	1.65	0.09
Bayon Lion	749k	1.42	6	1.55	-	-	-	0.70	26	4.31	1.30	100	15.36	2.49	43	5.57	5.99	0.08	3.79	0.18
Helmet Moustache	941k	2.04	9	2.89	-	-	-	0.74	57	11.15	2.03	100	19.66	3.31	38	6.14	24.66	0.14	5.56	0.25
Zeus	1.3m	2.47	11	3.86	-	-	-	1.17	58	15.91	2.34	100	27.21	4.11	100	22.68	30.40	0.20	7.19	0.35
Alfred Jacquemart	1.4m	3.33	5	3.79	-	-	-	1.67	43	16.21	3.03	100	30.33	5.44	51	12.78	8.88	0.14	8.07	0.35
POINT CLOUDS																				
Oil Pump	103k	0.07	9	0.12	-	-	-	0.04	15	0.22	0.10	100	1.27	0.19	55	0.56	0.17	0.01	0.31	0.02
Caesar Merged	388k	0.29	6	0.40	-	-	-	0.17	18	1.14	0.41	100	5.52	0.83	87	3.98	4.90	0.06	1.50	0.10
Truck	1.2m	0.99	17	3.20	-	-	-	0.65	26	5.53	1.27	100	18.87	3.67	72	10.77	5.63	0.14	5.09	0.29
Ignatius	1.4m	1.26	7	1.87	-	-	-	0.78	33	8.41	1.59	100	21.61	4.42	100	17.78	8.92	0.18	6.13	0.36

than 100k vertices, our approach is faster than Liu et al. for both the Poisson problem and data smoothing. The same holds for Shi et al. for the Poisson problem. For data smoothing, we are faster for one solve in 83% of cases and for three solves in 93% of cases. Compared to the Pardiso solver, we are faster for one solve of the Poisson problem in 92% of cases and for three solves in 95% of cases (data smoothing, 1x: 95%, 3x: 98%). Note, however, that our solver stops at a higher residual error than direct solvers. The strength of multigrid approaches is in settings where one needs a quick and relatively accurate solution. A direct solver is often preferable in settings where high accuracy is required.

Figure 4.6: All (non)manifold triangular meshes used in our experiments. Mosaic generated with code from Qingnan Zhou.

Figure 4.7: Residual over iteration count for Ours, Liu et al., and Shi et al. for data smoothing on Murex Romosus with $\alpha = 1 \times 10^{-3}$. See Figure 4.1 for a plot over time.

To provide insight into the convergence of our approach compared to the other GMG schemes, we plot convergence for a data smoothing problem with $\alpha = 1 \times 10^{-3}$ for the Murex Romosus shape in Figure 4.1 and the same plot against number of iterations in Figure 4.7. Again we see that our approach is on par with Liu et al. and beats Shi et al. with a high margin. More convergence plots for data smoothing, including plots over the number of iterations, can be found in the supplement. These confirm our results. There are some outliers: for Red Circular Box, Shi et al. converges faster than the other GMG approaches and for Moses, Gravo MG slows down around a residual of 1×10^{-6}.

4.5.5 Applications

We evaluate our solver in three scenarios: data smoothing, a geometric flow, and physical simulation. We compare solving times to a sparse Cholesky solver, commonly used for these problems.

Data smoothing For data smoothing, we consider an input function y on a surface and compute a smoother function x by minimizing a quadratic objective

$$(x - y)^\top M(x - y) + \alpha x^\top S x + \beta x^\top S M^{-1} S x. \tag{4.4}$$

The first term is a data term that penalizes deviation from the input function, the second and third terms are Laplace and bi-Laplace smoothing energies and $\alpha, \beta \in \mathbb{R}^{\geq 0}$ are parameters. Results are shown in Figures 4.8 and 4.9. The figures list timings for solving the

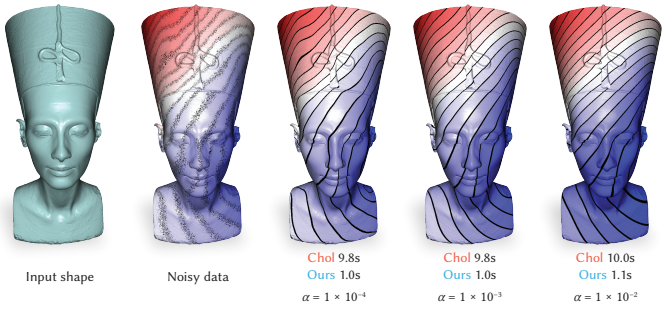


Figure 4.8: Smoothing of scalar data on a surface mesh with various parameter settings (Model: Nefertiti, 1m vertices) using the Dirichlet energy as smoothness energy.

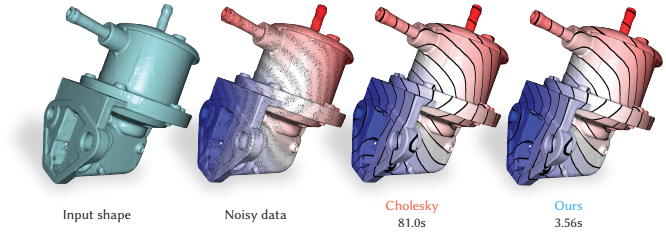


Figure 4.9: Smoothing of scalar data on a surface mesh (Model:Oilpump, 570k vertices) using a weighted sum of the Dirichlet energy and a bi-Laplacian energy as smoothness energy.

linear systems with our method and Eigen's sparse Cholesky solver. When changing parameter α to adjust the amount of smoothing, the direct solver needs to compute a new matrix factorization resulting in significant solving-time differences compared to our solver, in particular, when the bi-Laplacian energy is included.

Conformal flow As an example of a nonlinear geometric flow, we consider the conformal flow [108]. For robustness, we use an implicit time-integration that requires solving a linear Laplace system for every time step. We show results in Figures 4.10 and 4.11 and compare our solving times to those of Eigen's sparse Cholesky solver. Since the system matrix changes every time step, the direct solver constantly needs to compute new factorizations, resulting in substantial differences when performing multiple steps.

Balloon inflation As an example of a physical simulation, we consider the balloon inflation from [211]. A surface mesh represents a thin-layered rubber balloon that undergoes membrane deformation subject to air pressure. For time-integration an implicit Euler scheme is used and the resulting nonlinear equations are solved using a Newton scheme. To find the descent direction a sparse linear system is solved. As for the geometric flow, due the simulation's nonlinearity, the system matrix changes with every time step, forcing the direct solver to compute a new factorization in every time step. Results and timings are shown in Fig 4.12.

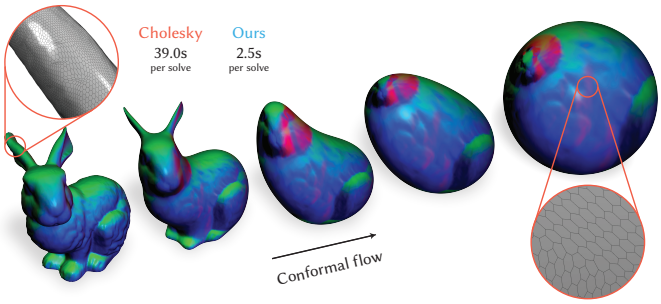


Figure 4.10: Conformal flow on polygon mesh. (Bunny model, 626k vertices).

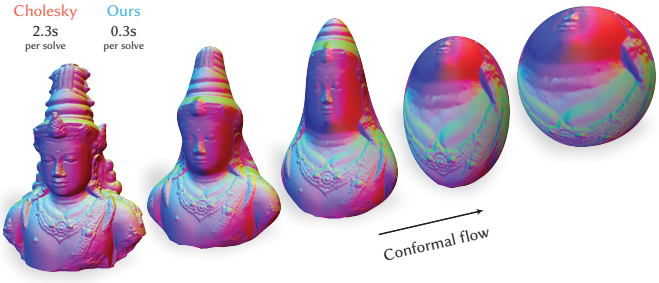


Figure 4.11: We compare the performance of our multigrid solver against a direct solver on conformal mean curvature flow on a nonmanifold mesh. (Indonesian statue model, 295k vertices).

Figure 4.12: Simulation of balloon inflation (Model: Armadillo, 180k vertices).

4.6 Conclusion

We introduce Gravo MG, a surface multigrid method that features fast hierarchy construction, applicability to general surface representations, and fast convergence. Our experiments demonstrate excellent performance compared to other GMG and AMG methods and direct solvers.

Conceptually, our method deviates from the common paradigm of GMG to represent levels via watertight meshes obtained by edge collapse. We use the geometry of the surface, while AMG ignores it for hierarchy construction. This opens a new direction for GMG on manifolds, which are generally applicable and fast to build, hereby improving the scalability of geometry processing methods.

In future work, graph Voronoi diagrams could be used for point cloud processing. We are excited about the quality of the triangles we generate from the graph Voronoi diagrams and see a potential use, when fast triangulations or uniform samples on point clouds are needed. Regarding theory, it would be interesting to explore under which conditions this approach can provide guarantees regarding the triangulation. Further acceleration is still possible. An important aspect is parallelization of both the hierarchy construction and the solver. Our solver could also become a preconditioner for a Krylov method like GMRES or CG to accelerate convergence.

5 Fast and Uncertainty-Aware SVBRDF Recovery from Multi-View Capture using Frequency Domain Analysis

*Relightable object acquisition is a key challenge in simplifying digital asset creation. Complete reconstruction of an object typically requires capturing hundreds to thousands of photographs under controlled illumination, with specialized equipment. The recent progress in differentiable rendering improved the quality and accessibility of inverse rendering optimization. Nevertheless, under uncontrolled illumination and unstructured viewpoints, there is no guarantee that the observations contain enough information to reconstruct the appearance properties of the captured object. We thus propose to consider the acquisition process from a signal-processing perspective. Given an object's geometry and a lighting environment, we estimate the properties of the materials on the object's surface in seconds. We do so by leveraging frequency domain analysis, considering the recovery of material properties as a deconvolution, enabling fast error estimation. We then quantify the uncertainty of the estimation, based on the available data, highlighting the areas for which priors or additional samples would be required for improved acquisition quality. We compare our approach to previous work and quantitatively evaluate our results, showing similar quality as previous work in a fraction of the time, and providing key information about the certainty of the results.**

* This chapter is based on the paper submitted as "Fast and Uncertainty-Aware SVBRDF Recovery from Multi-View Capture using Frequency Domain Analysis" and currently under review.

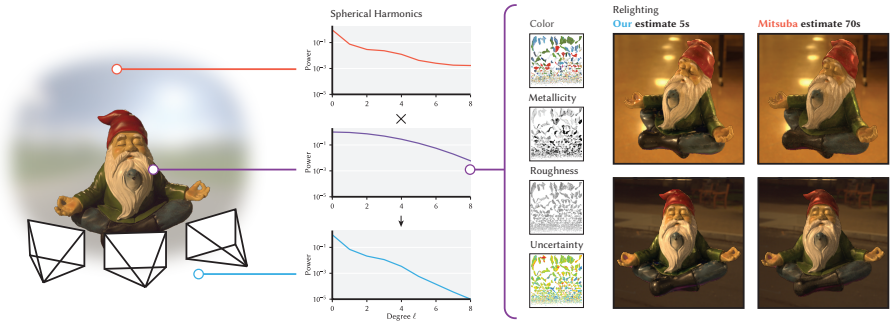


Figure 5.1: We present fast multi-view material acquisition for objects captured in uncontrolled setups (left). We propose to build upon and extend the signal processing framework for inverse rendering proposed by Ramamoorthi and Hanrahan [190]: we improve the model with shadowing and masking and propose a lightweight objective function for BRDF fitting using spherical harmonics power spectra (center). Using this objective, we propose an uncertainty estimation approach relying on statistical entropy. We show that our material estimation is significantly faster than previous work and achieves similar or better results.

5.1 Introduction

Object reconstruction is highly attractive for a variety of applications: from creating assets and environments for movies and video games, to preserving cultural-heritage objects digitally. Completely capturing the appearance properties of an object would require hundreds of photographs under controlled viewpoints and lighting conditions. Unfortunately, fine control of the lighting and viewing conditions is inconvenient, and impossible in many setups, such as outdoors or in a crowded museum. In such scenarios only a more "passive" capture of object appearance is possible. In this work, we therefore assume no control over lighting and suppose that views are captured in an unstructured manner. While this type of capture can be enough to recover the geometry of an object, it is more challenging to acquire accurate appearance properties.

Recent approaches for BRDF recovery from (under-constrained) multi-view capture can mainly be classified into two categories: (a) acquisition from only a few images, relying on deep network priors [44] and (b) optimizing directly for appearance parameters using differentiable rendering [102, 164]. The methods typically do not provide accuracy guarantees, and the latter ones optimize an ill-posed system, without providing any uncertainty measure. The optimization methods are often slow, due to requiring many iterations of full differentiable rendering for every view.

It is easy to miss information about the specular behavior of an object when lighting and viewpoints are uncontrolled, as it requires a lucky alignment of light sources and viewing directions. To shine a metaphorical light on this missing information, we propose to jointly estimate the object's SVBRDF parameters and the associated uncertainty (Figure 5.1). We improve upon a seminal work studying inverse rendering problems from a signal-processing

perspective in the frequency domain [190]. Moving to the frequency domain allows one to gain insight about the expected stability of an inverse rendering system to solve. For example, the incoming light must contain enough amplitude in high-frequency bands to recover certain roughness values for a microfacet BRDF. Otherwise, noise in the signal could make the inversion unstable. Without any amplitude, the inversion would be ill-conditioned, but this is unlikely to occur for natural lighting. Unfortunately, mapping the analysis in the frequency domain to practical algorithms is non-trivial: For instance, samples are often assumed to be equally spaced for frequency analysis while we deal with unstructured sparse samples. Moreover, in its original formulation, the conclusions from Ramamoorthi and Hanrahan [190] require a priori knowledge of the BRDF to conclude whether the inversion problem is well-conditioned.

In this work, we tackle these problems with an improved frequency-based formulation, providing both computational efficiency and insight into the underlying uncertainty. Our SVBRDF estimation is as accurate as differentiable rendering-based optimization and $13-35$ times faster than the state-of-the-art differentiable path tracer, Mitsuba 3 [102]. By finetuning our results with Mitsuba, we achieve better relighting results (+0.5dB PSNR) in less than a third of the time. Thanks to our extension of the signal processing framework, we obtain an accurate estimate of statistical entropy, interpreted as an uncertainty measure for each surface point. We show that this uncertainty can be used to improve acquisition by information sharing or providing information about the information provided by a given view.

To allow meaningful analysis or efficiency gains, we make the assumption that the geometry and lighting are estimated or captured using existing methods. For geometry we can use SDF extraction [195] or scanning [120], while lighting can be captured using a chrome ball or an omnidirectional camera. This is unlike recent inverse rendering methods, which assume all components are unknown [164, 219]. Our underlying goal is to make the best use of the available data and to better understand where signal is missing, before resorting to data-driven or ad hoc priors.

Following Ramamoorthi and Hanrahan, we model reflected light from a surface as a convolution of BRDF and incoming light on the hemisphere. To obtain the BRDF, we reconstruct this convolution filter and map it to analytical BRDF parameters. Working in the frequency domain, a convolution is a per-coefficient multiplication, which greatly simplifies analysis and computational complexity. We propose several key improvements to Ramamoorthi and Hanrahan's approach. First, we handle irregular light-field sampling by transforming the observations of both the outgoing and incoming light into spherical harmonics, using an efficient regularized least-squares

approach. Hereby, we can use sparse irregular data from the view sampling as well as regular Fibonacci samples from the environment lighting in a common representation. Next, we include the shadowing and masking terms of the BRDF, discarded by Ramamoorthi and Hanrahan, and update the spherical harmonics coefficients during a gradient-descent optimization. We obtain results on par with a full-featured differentiable rendering system [102]. We leverage the relationship between the spherical harmonic *power spectra* of incoming and outgoing light to define a new objective function for inverse rendering. As this objective is extremely lightweight to evaluate, both in terms of time and memory, we can evaluate the error for many parameter combinations in parallel. We interpret the error as a posterior distribution, encoding how likely each material parameter combination is, given the observations. This statistical perspective lets us compute the entropy of the posterior distribution as a measure of uncertainty. Intuitively, observations that can be well explained by many different parameter combinations lead to a 'spread-out' probability distribution with high entropy, which reflects a higher uncertainty. In contrast, when only a few parameter combinations are likely, the distribution is concentrated with low entropy, reflecting high certainty. Similarly, we can estimate the information gain from a new view by estimating its impact on entropy, potentially providing interactive guidance during capture. This uncertainty is also highly relevant to weigh priors, increasing prior weights for uncertain estimations.

We validate our material parameter- and uncertainty estimation in both synthetic and real acquisition conditions, comparing the estimation with recent methods. Our solution performs on par with state-of-the-art while being 13 times faster and providing uncertainty. Further, we validate our refined signal-processing framework through ablation studies, demonstrating clear improvement over Ramamoorthi and Hanrahan. In summary, we enable fast material reconstruction and uncertainty estimation via contributions to the space of SVBRDF estimation from multi-view captures under natural lighting:

- a frequency-space method using spherical-harmonics power spectra for efficient BRDF approximation and parameter exploration from sparse, irregular samples,
- we improve the convolution approximation of Ramamoorthi and Hanrahan by incorporating shadowing and masking,
- we quantify uncertainty in BRDF recovery by leveraging statistical entropy,
- we evaluate various applications for improved reconstruction based on fast acquisition and uncertainty.

5.2 Related Work

Capturing real-world, spatially varying, surface reflectance models from several camera views has been a long-standing challenge in computer graphics. The spatially varying bidirectional reflectance distribution function (SVBRDF) maps incoming light from a given direction to outgoing light observed at another direction and varies over six dimensions. Obtaining SVBRDFs from multi-view images has typically been addressed through optimization, using simplified models, and using data-priors trained on either realistic or synthetic SVBRDFs, or real-world measurements.

5.2.1 Optimization-based Capture

Various methods recover the associated material properties through optimization using a set of photographs of an object or surface. This task typically requires many photographs and controlled lighting [3, 165, 52] or object orientations [49] during acquisition to guarantee that specular effects are sufficiently observed. Some approaches rely on specialized hardware, for example to capture polarimetric information [99]. Others propose to rely on priors, such as stationarity of the captured materials [4, 2, 259, 88] to compensate for limited information. Multiple methods [164, 148, 171, 231] leverage recent progress in differentiable rendering [174, 102, 173, 260, 122, 170, 169, 172, 213, 257, 249, 27] to propose joint optimization of light, geometry, and material properties. Recent approaches build on novel representations for volumetric scenes, such as neural networks [161] and 3D Gaussians [110], to include optimization for material properties [15, 14, 55, 267, 215, 106, 266, 11, 150, 269, 248]. In general, these approaches cannot guarantee that the optimized results are accurate, as there is no guarantee that the provided photographs sample the necessary light-view angle pairs to qualify the specular behavior and are unable to provide any measure of certainty. Moreover, they rely on a heavy optimization process. Inspired by Ramamoorthi and Hanrahan [190], who describe the reflection function as a convolution in the frequency domain, we propose to leverage the efficiency of this framework to both estimate the material properties and their uncertainty, providing key information about which part of the material properties are likely faithfully reconstructed and for which parts we simply do not have enough information.

5.2.2 Data priors for Capture

Various approaches propose using data-based priors to simplify and enable low-information acquisition, allowing for estimating (SV)BRDFs from as little as a single image. Many such approaches

target flat surfaces, trained on a large amount of data using environmentally lit image(s) [131, 153, 229, 206] or flash-lit image(s) [43, 44, 45, 199, 275, 276, 274, 80] for acquisition. MaterialGAN [81] and Gao et al. [70] propose to optimize in latent spaces of deep neural networks, hereby remaining in the manifold of valid materials. In the context of 3D object acquisition, methods often focus on material extraction using a few (or even single) flash or multi-focal photographs [134, 46, 16, 60]. These approaches are orthogonal to our method, as we focus on recovering the SVBRDF from the provided signal without initial prior, quantifying the uncertainty of the process. Our uncertainty can in turn be used to better guide the use of priors to surface regions which most need it.

5.2.3 Frequency-based light transport

Our work builds on the frequency analysis of light transport, in particular with the idea that BRDFs can be expressed as low-pass filters [53, 190]. From this, one can express the reflection of light with a BRDF as a multiplication in the frequency domain. This idea is successfully used in the context of controlled illumination [73, 3], controlling the frequency of light patterns to estimate the BRDF filter parameter through a deconvolution of the reflected light. We do not assume control of the light and operate in the spherical-harmonics frequency domain rather than the Fourier domain [3] or custom basis functions [73]. This allows our analysis to work with arbitrary natural lighting environments. Closest to our approach is the work by Ramamoorthi and Hanrahan [190], which explicitly derives the concept of reflection as convolution in a signal-processing framework. They further outline the implications of this perspective on the well-posedness of BRDF- and light estimation from multi-view inputs and optimize for BRDF parameters through the frequency domain. Our work differs in a number of ways: First, we add theoretical insights on uncertainty and sampling. We propose methods to quantify these concepts, so they can be used in downstream tasks and show how to accelerate this approach using the power spectrum. Second, our approach supports arbitrary light setups and treats all frequencies in the same framework, rather than separating high and low frequency lighting. We do so by robustly estimating spherical harmonic coefficients directly on sparse, irregular samples using a regularized least-squares method. Finally, we propose to improve the BRDF model described by Ramamoorthi and Hanrahan to include shadowing and masking and show how to map the model to the widely used simplified principled BRDF [25].

5.2.4 Uncertainty estimation

Uncertainty estimation in the context of acquisition is highly desirable, as it can guide the capturing process and the use of priors,

or simply inform on the expected quality of a given reconstruction. Lensch et al. [129] estimate an object's material properties by clustering similar BRDF estimations within an object. To guide the clustering and its splitting, the covariance of the parameters are used. In Lensch et al. [130], the uncertainty of BRDF parameters is estimated using similar covariance matrices. More recently, Rodriguez-Pardo et al. [193], taking inspiration from Bayesian methods [69], use Monte-Carlo dropout to estimate the uncertainty of material estimation from a single picture. In the context of novel view synthesis, Goli et al. [74] propose to evaluate the inherent volumetric uncertainty of NeRF [161] reconstructions posterior to training. They use ray perturbations and an approximation of the Hessian to quantify uncertainty and show a correlation between uncertainty and absolute error. In this work, we use a different approach to estimate uncertainty. We use a fast BRDF estimation approximation for many parameter combinations and interpret the resulting error as a negative log-likelihood from which we derive an entropy measure.

5.3 Background

Our method builds on prior work in inverse rendering and spherical harmonics. We summarize required background knowledge and refer to related work for further depth.

5.3.1 Spherical Harmonics

Spherical harmonics are a series of orthonormal basis functions on the sphere, indexed by their degree ℓ and order m. We use them to represent incoming and outgoing radiance over incoming and outgoing directions on the unit sphere. Spherical harmonics are analogous to the Fourier series on a flat domain, where the frequency of the Fourier series corresponds to the degree ℓ and order m. We provide a brief overview of properties relevant to our method. For further details, a helpful reference and software package is published by Wieczorek and Meschede [243].

Any real, square-integrable function on the sphere can be expressed as a spherical-harmonics series:

$$f(\theta,\phi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} f_{\ell m} Y_{\ell m}(\theta,\phi), \tag{5.1}$$

where $f_{\ell m}$ is the coefficient for spherical harmonic $Y_{\ell m}(\theta,\phi)$,

given as

$$N_{\ell m} = \sqrt{\frac{(2-\delta_{m0})(2\ell+1)}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} \tag{5.2}$$

$$Y_{\ell m}(\theta,\phi) = \begin{cases} N_{\ell m} P_\ell^m(\cos\theta)\cos m\phi & \text{if } m \geq 0, \\ N_{\ell m} P_\ell^{|m|}(\cos\theta)\sin|m|\phi & \text{if } m < 0. \end{cases} \tag{5.3}$$

$N_{\ell m}$ is a normalization factor, δ_{m0} is the Kronecker delta function, which evaluates to 1 when $m = 0$, and P_ℓ^m is the associated Legendre function for degree ℓ and order m. The total number of spherical harmonics up to- and including a maximum degree, ℓ^*, equals $(\ell^*+1)^2$. A useful property of spherical harmonics in our setting is that a rotational convolution on the sphere is equal to multiplication of coefficients in the spherical harmonic domain.

The power spectrum of a spherical function f can be computed from the spherical harmonic coefficients per degree

$$S_f(\ell) = \sum_{m=-\ell}^{\ell} f_{\ell m}^2. \tag{5.4}$$

The power spectrum is invariant to rotations of the coordinate system. In our context that means the power spectrum is invariant to slight perturbations of the normals at each point.

Computing spherical harmonic coefficients One can find the SH coefficients for a function f by computing the inner product with the basis functions

$$f_{\ell m} = \int_{S^2} f(\theta,\phi) Y_{\ell m}(\theta,\phi) d\omega. \tag{5.5}$$

A useful property holds for the coefficient of degree $\ell = 0$, for which the spherical harmonic is constant; $Y_{00}(\theta,\phi) = (4\pi)^{-\frac{1}{2}}$. The corresponding coefficient, f_{00}, is equal to the integral of f times the normalization constant, $(4\pi)^{-\frac{1}{2}}$. The spherical harmonics for higher degrees all integrate to zero[1]. This is relevant in the context of rendering, because the total integrated incoming and outgoing radiance can be read from the 0^{th} degree coefficient and that coefficient alone. In the general case, we estimate the coefficient for $f_{\ell m}$ based on samples of f. The sampling method determines how these coefficients are estimated.

1: Because the spherical harmonics are orthormal, the inner product between any spherical harmonic with $\ell > 0$ and the constant function ($\ell, m = 0$) equals zero.

Regular sampling If f is sampled on a grid with equally spaced longitudinal and latitudinal angles, Equation 5.5 can be accelerated using a fast Fourier transform in the longitudinal direction ϕ and a quadrature rule in the latitudinal direction θ [51]. This approach can be used for environment maps represented as rectangular textures.

Irregular sampling During capture the camera is often placed at irregular positions, leading to non-uniform (θ, ϕ) samples. Further, a point on the surface might be observed from only a few positions. We therefore often need to use sparse and irregular samples to fit spherical harmonic coefficients. We do so by fitting the coefficients using least-squares, expressing Equation 5.1 as a linear system

$$\mathbf{Yc} \approx \mathbf{f}, \tag{5.6}$$

where $\mathbf{f}$ is a vector of n discrete samples from f, $\mathbf{Y}$ is a matrix of size $n \times (\ell^* + 1)^2$ containing the spherical harmonics sampled at the same locations as f, and $\mathbf{c}$ is a vector of the $(\ell^* + 1)^2$ coefficients we want to find. We can find $\mathbf{c}$ by solving a least-squares system

$$\mathbf{Y}^\top \mathbf{Yc} = \mathbf{Y}^\top \mathbf{f}. \tag{5.7}$$

To be well posed, this system requires $n > (\ell^* + 1)^2$ independent samples, which can be challenging in the context of sparse sampling, making the system under-constrained. We propose to use a custom regularizer in Section 5.4.2 for cases where the number of samples is too low.

5.3.2 *Reflection as Convolution*

Surface reflection can be approximated as the convolution of incoming radiance (from the light direction) with a BRDF [190]. Specifically, if we assume an isotropic microfacet Torrance-Sparrow BRDF, combined with a Lambertian term, we can derive the following approximate equation for outgoing radiance B at point p in the view direction ω_o

$$B(p, \omega_o) \approx K_d E(p) + K_s F(\theta_o) \left[S_\alpha * L(p) \right]_{\omega_o}, \tag{5.8}$$

where K_d and K_s are diffuse and specular terms; $E(p)$ is the irradiance integrated over the hemisphere; $F(\theta_o)$ is a simplified Fresnel term, which only depends on the outgoing direction; $L(p)$ is the incoming radiance; and S_α is a filter parametrized by the distribution width, α. This filter is derived from the normal distribution function of the surface. The $*$ operator represents convolution. A derivation of this approximation is included in the Appendix C.

In this framework, estimating the specular BRDF parameters comes down to estimating the convolution kernel. This can be done efficiently since a convolution in the angular domain can be represented as a multiplication in the spherical harmonics frequency domain. Using this representation, one can find the convolution kernel through a division of the spherical harmonics coefficients of the outgoing radiance by those of the incoming radiance. This is analogous to kernel estimation for image deblurring in the Fourier domain.

The above leads to crucial insights regarding the well-posedness of BRDF recovery. Ramamoorthi and Hanrahan state that the recovery of BRDF parameters is ill-posed if the input lighting has no amplitude along certain modes of the filter (BRDF). Those modes cannot be estimated without additional priors on plausible spatial parameter variations. For the microfacet BRDF, this leads to the following conclusion: if the incoming light only contains frequencies $\ell \ll \alpha^{-1}$, multiplying the coefficients of the light with those of the BRDF only results in a small difference, and the inversion of this operation is ill-conditioned. To accurately estimate α, the incoming light used during the capture needs to exhibit sufficiently high frequencies.

This insight is based on a derivation for the coefficients of the microfacet model. The normalized SH-coefficients of the specular component of the BRDF for normal incidence, S, are approximated by

$$\hat{f}_{\ell m} \approx e^{-(\alpha \ell)^2}, \tag{5.9}$$

which is a Gaussian in the frequency domain with a width determined by α. The kernel is derived from a Beckmann normal distribution function and the α parameter corresponds to the α parameter there. Note that these coefficients do not vary with the Spherical Harmonics order m, since the normal distribution function is isotropic for outgoing rays in the direction of the normal vector ($\theta_o = 0$). An important approximation employed by Ramamoorthi and Hanrahan is that this same kernel can be used for any outgoing direction. While the correct kernel varies with the incoming and outgoing direction, this approximation does not lead to significant error for inverse rendering [190] as they note that *"it can be shown by Taylor-series expansions and verified numerically, that the corrections to this filter are small* [for low degrees ℓ]".

In this chapter, we expand on the theory established by Ramamoorthi and Hanrahan by improving the reflection as a convolution model's accuracy and developing the implications for well-posedness into quantifiable metrics on uncertainty without a-priori knowledge on α.

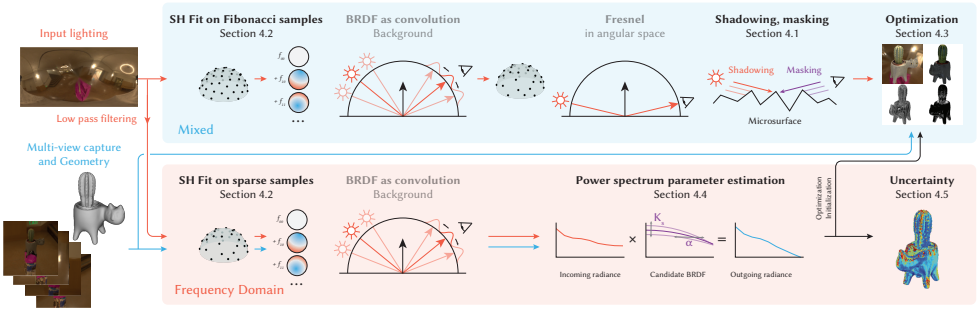


Figure 5.2: An overview of our pipeline showing input, output, and the proposed algorithm. The input to our approach is a set of photographs of an object from multiple camera points and the associated camera extrinsics and intrinsics. We also provide the input lighting as an environment map and object geometry. Next, we propose two main variants of our approach: a mixed pipeline (top) and a pipeline fully contained in the spherical harmonic domain (bottom). **The Mixed pipeline** first estimates spherical harmonic coefficients for the input lighting on Fibonacci samples. The light is convolved with the BRDF filter and mapped back to the angular domain, where the outgoing radiance is attenuated with Fresnel, shadowing, and masking. During acquisition, we optimize the BRDF parameters by comparing the outgoing light samples to the radiance captured in the input photographs. **The Frequency Domain pipeline** first estimates spherical harmonic coefficients on both the incoming and outgoing radiance and then estimates the effect of different BRDF filters within the power spectrum. This is used to compute a measure of uncertainty on the predicted parameters for acquisition.

5.4 Method

Our goal is to recover (SV)BRDF properties and quantify uncertainty from multi-view capture of an object under environment lighting. The input to our method is a set of images $\mathcal{I}$ captured from multiple camera positions. The camera extrinsics, intrinsics, object geometry and HDR lighting are assumed to be known – through existing methods for camera calibration, photogrammetry, and HDR environment capture – but not controlled.

For each point p on the object surface, we know the incoming radiance $L(p, \omega_i)$ from directions ω_i. These values can be sampled from the environment map. For higher accuracy, one can attenuate the light based on self-occlusion, but we do not explore this in our work. By projecting the captured images onto the surface, we retrieve the outgoing radiance at each point, $B(p, \omega_o)$. Our task is to recover the BRDF $f(p, \omega_o, \omega_i)$ and capture the uncertainty associated with its parameters. The BRDF relates the incoming radiance of the upper hemisphere Ω to the outgoing radiance as [182]:

$$B(p, \omega_o) = \int_\Omega f(p, \omega_o, \omega_i) L(p, \omega_i) \cos\theta_i d\omega_i. \tag{5.10}$$

We use the Torrance-Sparrow BRDF model with parameters for diffuse reflectance K_d, specular reflectance K_s, and the slope of the normal distribution function α (Equation 5.8). We denote these parameters as a vector of parameters, ψ. In subsection 5.4.6, we show how to map a simplified Disney principled BRDF [25] to these parameters.

To estimate ψ, we propose to build on the framework proposed

by Ramamoorthi and Hanrahan [190]. In the following sections we carefully highlight how we improve and extend its use compared to the original formulation. An overview of the pipeline is shown in Figure 5.2, where we denote our contributions in black and existing work in gray. In particular, we describe the extension of the existing convolution model to take shadowing and masking effects into account. We then describe an efficient spherical-harmonics fitting from sparse and irregular samples of radiance, enabling unstructured capture setups. We then propose a carefully validated approximation of the convolution model to design a lightweight loss function, leveraging the power spectrum for efficient loss evaluation. These improvements enable quick sampling of the BRDF parameter space, Ψ, which we use to propose a new formulation for capturing uncertainty relying on statistical entropy. Finally, we map the Torrance-Sparrow BRDF model parameters to those of the simplified Disney principled BRDF [25] for use in modern rendering pipelines.

5.4.1 Improving the Convolution Model

The approximate reflection function in Equation 5.8 does not include the shadowing or masking terms present in microfacet models [182]. The shadowing and masking terms model occlusion on incoming light (shadowing) and outgoing light (masking) due to the configuration of the microfacets (Figure 5.2, top right). Light at grazing angles is more likely to be occluded by microfacets, especially if the microfacet distribution is wide (high roughness). Ramamoorthi and Hanrahan argue that these terms can be ignored, because they mostly affect observations made at grazing angles. While this is true for materials with low roughness, we find that ignoring this term leads to reconstruction errors for high roughness materials – we evaluate the terms' impact in Table 5.4. We therefore propose to introduce shadowing and masking terms to the convolution model.

Shadowing and masking effects are typically modeled jointly to avoid an over-correction of the radiance (Pharr et al. [182], Section 9.6.3). However, this joint term cannot be included in the simplified convolution model as presented in Equation 5.8, because the kernel would need to vary with both ω_i and ω_o (the 2D kernel only varies with either ω_i or ω_o). Therefore, we assume that shadowing and masking effects are independent and can be modeled separately as $G_\alpha(\omega_i)G_\alpha(\omega_o)$. While we do observe a small overestimation of the shadowing and masking effect, we show in Table 5.4 that the addition of the term still leads to improvements for BRDF acquisition. This way we can first attenuate the incoming light with the shadowing term, then convolve with the BRDF and then attenuate the result

with the masking term

$$B(p, \omega_o) \approx K_d E(p) + K_s F(\theta_o) G_\alpha(\theta_o) \left[S_\alpha * G_\alpha(\theta_i) L(p) \right]_{\omega_o}, \quad (5.11)$$

where G_α is the shadowing-masking function in the Trowbridge and Reitz [226] model (also referred to as GGX [233]). The shadowing and masking terms both depend on α, which is not known a-priori. This means that the shadowing and masking terms change as we optimize α. As a consequence, we need to update the coefficients for the incoming light as $G_\alpha(\theta_i)$ changes. While the estimation of spherical harmonic coefficients is relatively fast, it can still be quite cumbersome to update the coefficients in every optimization step. Therefore, we only update the shadowing term each m iterations.

5.4.2 Fitting Spherical Harmonic Coefficients

Much of our computation relies on a convolution applied in the spherical harmonics domain, which means we require a good spherical harmonics transform. This is challenging, because the outgoing light field, B, is sampled sparsely and non-uniformly (Figure 5.3, second column). Therefore, Ramamoorthi and Hanrahan do not directly estimate spherical harmonic coefficients on B, but only perform the transformation from the spherical harmonic domain to the directional domain. This is simpler, because it only requires evaluating the spherical harmonics at the sample locations (Equation 5.1). For the incoming light, they represent low-frequency (area sources) and high-frequency lighting (point sources) separately and limit their environments to controlled 'lightbox'-like settings. We would like to compute an estimate of the spherical harmonic coefficients for the outgoing radiance, because this would enable algorithmic analysis fully within the frequency domain. This is highly appealing, because it has the potential to greatly simplify the analysis of BRDF recovery, as showcased by Ramamoorthi and Hanrahan.

Next to sparse sampling, a significant challenge is that both the incoming- and outgoing radiance fields are only supported on the upper hemisphere when considering reflection. If we were to consider incoming radiance on the lower hemisphere, we would implicitly model light 'leaking through' the surface. One possible solution is to simply pad the lower hemisphere with zeros (Figure 5.3, third column). This is not practical, because it introduces high frequency variation at the boundary, which disrupts any potential analysis on the spherical harmonic coefficients (bottom row). Another solution that we explored is to represent the radiance fields with hemispherical harmonics [272]. In practice, this is equivalent to mirroring the radiance fields along the meridian, which resulted in mismatches with the convolution model.

We instead propose a spherical harmonics fitting approach that is robust to sparse and irregular samples, and supports the lack

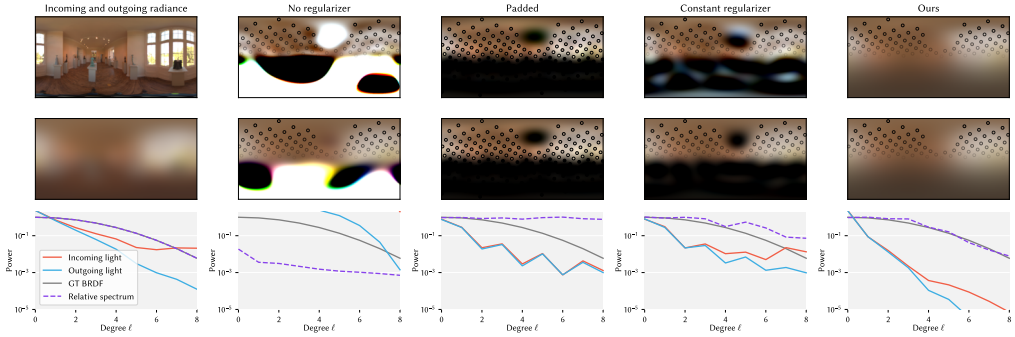


Figure 5.3: A comparison of the options for fitting spherical harmonics on sparse samples. In the first column, top: incoming radiance (synthetic) on the sphere mapped to a lat-long grid. Middle: synthesized outgoing radiance as a result of filtering incoming radiance with Equation 5.9, $\alpha = 0.2$. Bottom: power spectra of incoming and outgoing radiance and ground-truth filter (BRDF) plotted on a log-scale. In the second to fifth column, we show the result of fitting spherical harmonics coefficients to samples and then transforming back to the spatial domain. We use 100 samples from the upper hemisphere of the input radiance (first column) and simulate missing samples due to occlusion or missed camera positions by masking some points, leaving 88 samples. The samples are weighted with $\cos\theta$, which is visualized as the transparency of the samples. We find that our method with a weighted regularizer is able to smoothly interpolate missing values and retrieves the correct BRDF filter in the power spectrum, where other variants overfit, or add dark regions in the upper hemisphere.

of lower hemisphere samples and gracefully handles occluded regions. First, analog to ideas from compressed sensing, we add a regularization term on the spherical harmonic coefficients. A typical choice for this term is an $L1$ norm, which enforces sparsity in the spherical harmonic coefficients. The resulting system can be solved with linear programming. We propose instead to use a weighted $L2$ norm which can be solved using standard linear solvers, which we found to be faster and more stable for our usecase:

$$(\mathbf{Y}^\top \mathbf{Y} + \lambda \mathbf{W})\mathbf{c} = \mathbf{Y}^\top \mathbf{f}, \tag{5.12}$$

where $\mathbf{W}$ is a diagonal weight matrix. If constant, this weight matrix leads to poor fitting as the regularization is too strong on the lower spherical harmonics degrees (Figure 5.3, fourth column). We set the weight equal to e^ℓ, increasing the strength of regularization for higher spherical harmonics degrees. This is informed by the observation that many natural images have a power spectrum with exponential decay [65]. Intuitively, this regularizer encourages filling unknown regions with low-frequency information, akin to solutions with a smoothness term (Figure 5.3, right-most column). By applying the same regularization to both incoming- and outgoing radiance fitting, we are able to recover the correct filter (bottom row), even though the recovered incoming radiance looks blurrier (top row).

We fit the signals for which we have complete spatial information (e.g., incoming environment light) by sampling it in with Fibonacci samples on the sphere. For partially observed signal (e.g., reflected light) we use the available samples with the described fitting technique for sparse and irregular samples. An important consideration

related to the number of samples, is the frequency of the signal that we fit spherical harmonics to. We have an in-depth analysis of this in Appendix C. The practical take-away is that the incoming signal should be bandlimited to roughly $n^{\frac{1}{2}}$ to avoid aliasing, where n is the number of input views. We ensure this is the case for the incoming radiance by applying the filter in Equation 5.9 with $\alpha' = n^{-\frac{1}{2}}$ on the input environment map. The impact on our BRDF estimation is that we cannot accurately recover $\alpha < \alpha'$. In our experiments and analysis in Appendix C, we find that the recovered α tends to be between 0 and α' in those cases. A second finetuning pass with a differentiable path tracer, like the one we demonstrate in the experiments, can help refine those regions.

5.4.3 Optimizing for SVBRDF Parameters

Using our more complete model described in Equation 5.11, we can optimize for the object's material parameters using gradient descent:

$$\psi^* = \arg\min_{\psi} \sum_{p,\omega_o} |B_\psi(p, \omega_o) - B'(p, \omega_o)| \tag{5.13}$$

With $B_\psi(p, \omega_o)$ a rendering operator using Equation 5.11 and B' the radiance values projected onto the surface points p. As described in subsection 5.4.1, we optimize the parameters ψ for a number of iterations before updating the shadowing term, as this step requires recomputing the spherical harmonic fit. The masking term is included in the forward rendering step and updated every iteration. Because of the simplicity of the convolution approximation, we are able to optimize for all points and camera positions in one go, rather than using stochastic gradient descent with batches of rays. This lets us run our optimization with fewer iterations than other approaches. Similar to other methods, we add the total variation norm on the parameters in texture space as a regularizer to enforce smoothness, weighted with 1×10^{-3}.

5.4.4 A Lightweight Objective

In this section we define a particularly efficient-to-compute approximation of the convolution BRDF model described in Equation 5.11. Our goal is to develop algorithms to analyze the inverse rendering problem fully in the spherical harmonic domain. This would simplify such analysis tremendously, as convolution is simply a per-frequency multiplication in this domain.

For this approximation, we propose to ignore the Fresnel and shadowing and masking terms, letting us express the reflection function entirely in terms of spherical harmonic coefficients. We

show experimentally in Section 5.5.4 that the impact of this approximation is acceptable on the underlying application of this objective. Using Equation 5.9 for the coefficients of S we get:

$$B_{\ell m} = \begin{cases} \pi^{-\frac{1}{2}} K_d E(p) + K_s L_{\ell m}, & \text{if } \ell = 0. \\ K_s e^{-(\alpha \ell)^2} L_{\ell m}, & \text{otherwise.} \end{cases} \tag{5.14}$$

We note that the specular component for $\ell = 0$ does not depend on α, because for $\ell = 0$, Equation 5.9 equals 1. The term $\pi^{-\frac{1}{2}} K_d E(p)$ represents the diffuse component as a constant function. We know, based on the conservation of energy, that the diffuse component should integrate to $K_d E(p)$ on the upper hemisphere for outgoing directions. Thus, the integral over the full sphere should be $K_d 2E(p)$. Equation 5.5 shows that the Spherical Harmonics coefficient for degree $\ell = 0$ should be equal to $(4\pi)^{-\frac{1}{2}}$ times the integral: $(4\pi)^{-\frac{1}{2}} K_d 2E(p) = \pi^{-\frac{1}{2}} K_d E(p)$. From this expression, we make two observations. (a) We cannot recover the ratio between K_d and K_s from the $\ell = 0$ alone, as we have two unknowns and only one coefficient and (b) we cannot derive any information on α from the 0th degree, as it has no impact there. It follows that we can only recover K_s and α from degrees $\ell > 0$. Once K_s is known, we can then estimate K_d from the 0^{th} degree. In other words, we know diffuse reflectance once we know the contribution of the specular component. This reduces our analysis of uncertainty to the specular component on the parameters K_s and α. Note that this simplification comes naturally in the frequency domain, because we can simplify limit our analysis to degrees $\ell > 0$. This is non-trivial to separate out in the directional domain.

We use our proposed spherical harmonic coefficient fitting described in Section 5.4.2 to recover $L_{\ell m}$ and $B_{\ell m}$. $L_{\ell m}$ is estimated by sampling the known environment lighting L with the Fibonacci sampling, while $B_{\ell m}$ is estimated from the sparse, irregular samples of the object from the input views as described in Section 5.4.2.

Next, we propose to use the power spectra S_L and S_B of the spherical harmonics fittings $L_{\ell m}$ and $B_{\ell m}$, which we can express as

$$S_L(\ell) = \sum_{m=-\ell}^{\ell} L_{\ell m}^2 \tag{5.15}$$

$$S_B(\ell) = \sum_{m=-\ell}^{\ell} (K_s e^{-(\alpha \ell)^2} L_{\ell m})^2 \tag{5.16}$$

$$= K_s^2 e^{-2(\alpha \ell)^2} \sum_{m=-\ell}^{\ell} L_{\ell m}^2 \tag{5.17}$$

$$= K_s^2 e^{-2(\alpha \ell)^2} S_L(\ell) \tag{5.18}$$

for degrees $\ell > 0$. Using this relationship between the BRDF pa-

rameters (K_s, α) and the incoming and outgoing radiance power spectra ($S_L(\ell)$ and $S_B(\ell)$, we can formulate a lightweight objective:

$$D(K_s, \alpha) = \sum_{\ell=1}^{\ell^*} (S_B(\ell) - K_s^2 e^{-2(\alpha\ell)^2} S_L(\ell))^2. \tag{5.19}$$

Intuitively this objective evaluates the difference between the observed radiance and the BRDF-convolved incoming radiance with a given (K_s, α) through their respective power spectrum. This formulation reduces the computation from ℓ^{*2} to ℓ^* evaluations of the objective, making its evaluation near instantaneous. We show how this fast estimation is key to unlocking near-instant (<1ms) uncertainty evaluation as shown in Section 5.4.5 and can be used as initialization of the optimization described in Section 5.5.6.

5.4.5 Entropy as Uncertainty

In our context of passive acquisition, we have no guarantee that the illumination on the captured object surface is sufficient to fully recover the material parameters. This uncertainty is particularly desirable information as it is key to understanding the quality of acquisition (e.g. for digital twins), driving the use of priors in uncertain regions while preserving the correctly recovered surface areas, or driving the acquisition by maximizing the information provided by new views when additional captures are possible. We propose to use a grid search of the parameter space to inform our uncertainty estimation. This is now tractable, thanks to the power spectrum approximation discussed in the previous section. By interpreting the error over the parameter space as a likelihood function, we can express uncertainty using entropy, which is a common way to study uncertainty. We detail these contributions in the following paragraphs.

One can interpret the recovery of the right BRDF parameters, ψ^* as a maximization of the posterior probability distribution of the parameters given the incoming and outgoing power spectrum

$$\psi^* = \arg\max_{\psi} p(\psi | S_L, S_B). \tag{5.20}$$

According to Bayes rule, the posterior is proportional to the product of the likelihood and prior over the parameters:

$$p(\psi | S_L, S_B) \propto \mathcal{L}(S_L, S_B | \psi) q(\psi), \tag{5.21}$$

Assuming $q(\psi)$ to be an uninformative prior, i.e. a uniform distribution on the range of allowed parameter values (typically [0,1] for BRDF parameters), we can reformulate this objective to an

equivalent negative log-likelihood objective

$$\psi^* = \arg\min_{\psi} -\log(\mathcal{L}(S_L, S_B|\psi)). \tag{5.22}$$

We interpret the lightweight objective described in Equation 5.19 as the Negative Log Likelihood (NLL) of the joint probability, given parameters $\psi = (K_s, \alpha)$:

$$\mathcal{L}(S_L, S_B|\psi) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2} D(K_s,\alpha)}, \tag{5.23}$$

where $D(K_s, \alpha)$ is the squared error measure defined in Equation 5.19. We can view this as assuming that the *error* of our model caused by approximations and measure noise is Gaussian $\sim \mathcal{N}(0, \sigma^2)$. We set $\sigma = 1e^{-2}$, found empirically to capture the change in loss observed when noticeably changing ψ. The scaling constant above is optional since we will renormalize below in Equation 5.24. With σ fixed, minimizing the NLL from Equation 5.22 is equivalent to minimizing $D(K_s, \alpha)$.

We explore the material parameter space for $\psi = (K_s, \alpha)$, discretized in n values in total, $\psi_{i=0..n}$ ($\sqrt{n}$ per variable). Because this process can be parallelized easily, this step typically takes between $1 - 10$ms for common scenarios. The grid search provides us with a discrete distribution d obtained from the n samples of ψ:

$$d_i = \frac{\mathcal{L}(S_L, S_B|\psi_i)}{\sum_{i=1}^{n} \mathcal{L}(S_L, S_B|\psi_i)}, \tag{5.24}$$

where d_i gives the probability that a parameter is within a range of $1/2n$ from ψ_i. We use the discrete distribution to compute the uncertainty with the distribution's normalized entropy[2]:

$$H = -\frac{1}{\log(n)} \sum_{i=1}^{n} d_i \log(d_i). \tag{5.25}$$

Intuitively, the normalized entropy describes the spread of a probability distribution: low entropy means that the probability distribution is highly concentrated, which implies certainty. High entropy means that the probability distribution is spread out, which implies uncertainty: many options share a similarly high probability.

We show three examples of power spectra, their corresponding likelihood and entropy in Figure 5.4. The top row shows incoming light for a dirac delta light source (constant in the spherical harmonic domain) and we see that entropy is low ($H = 0.25$) and that it is simple to recover the correct parameters. The middle and bottom rows are problematic cases. In the middle row, the light only has amplitude in low frequencies and many roughness values are equally likely ($H = 0.69$), in line with the conclusions from Ramamoorthi and Hanrahan. The bottom row shows a material with very low

2: This is equivalent to computing entropy on a continuous probability density function using the limiting density of discrete points [103].

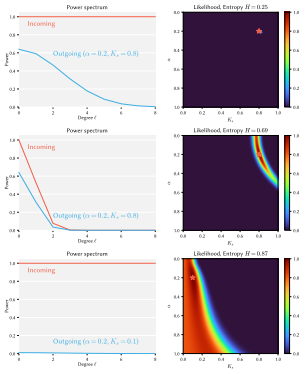


Figure 5.4: Examples of the likelihood and entropy H resulting from three sets of power spectra for incoming and outgoing radiance. The likelihood is sampled at 100 discrete values. **Top**: An ideal situation, where the incoming radiance is a dirac delta (all frequencies are present). **Middle**: A situation where the incoming radiance does not have enough amplitude for higher degrees. **Bottom**: A situation where the incoming radiance is ideal, but the specular component is too low to get a proper recovery for α.

specular reflectance, resulting in high entropy ($H = 0.87$). This could result in incorrect estimates under noisy conditions. Concluding, entropy as uncertainty generalizes and quantifies the observations made by Ramamoorthi and Hanrahan on the uncertainty for certain incoming light and material parameters, without requiring a prior estimate of the BRDF parameters.

5.4.6 Disney Principled BRDF

Many modern rendering pipelines employ variants of the Disney BRDF [25], which is a combination of a diffuse term and a microfacet term with a user-friendly parametrization. The model also contains some additional features beyond the scope of the current work. We can formulate our model using the principled BRDF parameters, rather than the raw parameters of the Torrance-Sparrow model. We parameterize base color, metallicity and roughness, mapping these to the Torrance-Sparrow model as

$$K_d = R_b, \tag{5.26}$$
$$K_s = 1, \tag{5.27}$$
$$R_0 = 0.04 + (R_b - 0.04)m, \tag{5.28}$$
$$F(\theta_o) = R_0 + (1 - R_0)(1 - \cos\theta_o)^5, \tag{5.29}$$
$$\alpha = r^2, \tag{5.30}$$

where R_b is the base color, m is the metallicity parameter and r is the roughness. In the ablations where the Fresnel term is not used, we set $K_s = R_0$. Note that we set K_s to 1, because the specular term is contained in the Fresnel term as R_0.

5.5 Experiments

Our method yields two main benefits: fast fitting and a measure of uncertainty. In our experiments, we validate these benefits with comparisons, provide insight into variations of the algorithm in ablations, and show applications to demonstrate potential use cases.

5.5.1 Implementation

We implement our full pipeline in PyTorch. The output of our method is a set of PBR textures (roughness, metallicity, base color) which are mapped to the input mesh with UV-coordinates to 512x512 textures. For re-renders, we feed the textures produced by our method into Mitsuba, which is possible due to the mapping to the principled BRDF described in subsection 5.4.6. Each timing result is reported on a machine with an NVIDIA RTX4090 GPU and an AMD Ryzen 9 7950X 16-Core CPU.

Our main comparison target is Mitsuba 3 [102], as it is a well-documented, open-source, and research-friendly package for differentiable rendering. More recently Sun et al. [219] propose to jointly optimize for lighting, material and geometry, but did not yet release a public implementation; given that they rely on a complex differentiable renderer, we believe Mitsuba 3 to be the best proxy for their inverse PBR step. When optimizing with Mitsuba, we sample one ray per pixel in each view. The material parameters are optimized with stochastic gradient descent, where the gradient for each step is estimated with a random set of rays to ensure good convergence. Each primary ray is sampled 32 times, allowing the reflection directions to be sampled. We run an Adam optimizer until convergence or a maximum of 10 epochs.

5.5.2 Datasets

We perform experiments on a recent in-the-wild benchmark dataset, Stanford ORB [120] and on synthetic scenes. Stanford ORB contains 14 objects, each captured 3 times in different scenes (lighting environments), selected from a total of 7 scenes. The lighting environment is captured through a chrome ball and stored as a lat-long environment map. Each object is also scanned in a separate stage, providing high-quality geometry. For our evaluation, we focus on the (SV)BRDF recovery step and use the provided geometry. In our experiments, we optimize material parameters for each object, for each of the three environments, and test against the photographs of the same object in the two environments that were not observed during optimization.

For the synthetic benchmark, we selected 15 objects with spatially varying BRDF textures for base color, roughness and metallicity. We selected four environment maps with varying challenges to render the objects: two indoor scenes, one outdoor scene with a clear sky and sun, and one overcast outdoor scene. Because we have ground-truth material textures for the synthetic scenes, we can quantitatively evaluate our optimization results and validate uncertainty directly on the optimized parameters. The objects are rendered in Mitsuba at 512×512 resolution and 256 samples per pixel. Renders and ground truth views for both Stanford ORB and the synthetic benchmark are included in the Supplement.

5.5.3 Acquisition comparison

We validate that our improved convolution model results in high-quality acquisition results. We also quantify the benefit of our method compared to other methods regarding optimization time. We test our approach on Stanford ORB and synthetic scenes.

Table 5.1: Benchmark Comparison for **Novel Scene Relighting** from [120]. All the models listed in this table were trained with the ground-truth 3D scan and ground-truth, in-the-wild environment maps. We report evaluations on the same dataset and task under different acquisition conditions in the Appendix.

	PSNR-H↑	PSNR-L↑	SSIM↑	LPIPS↓	Time
NVDiffRec [164]	24.319	31.492	0.969	0.036	142.143s
Mitsuba [102]	26.595	**34.185**	**0.977**	0.032	69.96s
Ours ‡	**26.677**	33.936	**0.977**	**0.029**	5.27s
Ours - Spectrum Only	24.478	31.117	0.971	0.037	**1.96**s
Ours + Mitsuba (1 epoch)	27.038	34.560	**0.978**	0.030	19.87s
Ours + Mitsuba (2 epochs)	**27.199**	**34.770**	**0.978**	**0.029**	33.39s

Stanford ORB The similarity metrics for relighting results on Stanford ORB are presented in Table 5.1. We find that our approach performs on par with Mitsuba, while demonstrating a 13 times speedup with little optimization on our side (e.g. we do not implement custom CUDA kernels and use pure PyTorch). We also compare to NVDiffRec, as it leverages a differentiable rasterizer, and show that our method achieves better relighting results. For NVDiffRec, given that we optimize for materials only, we monitor the loss and record the time for good convergence. We empirically find 2000 steps to be sufficient for this dataset. We qualitatively compare our results to Mitsuba and photographs of the objects under the same illumination in Figure 5.5 and against NVDiffRec in supplemental material. We present a view of our material channels projected on the captured object and a few input photographs. The entropy is visualized using the "Turbo" colormap in Matplotlib. On the four right columns, we present renderings of our results and Mitsuba under a novel illumination and compare them against photographs of the object taken under that same illumination. We can see that our approach recovers similar appearance in a fraction of the time.

Synthetic For synthetic datasets, we compare the recovered material parameters with the ground-truth parameters. We show the results in Table 5.2, demonstrating comparable accuracy compared to Mitsuba despite particularly challenging objects for our method (e.g. with significant inter-reflections – metallic chess pieces and self-occlusion – donut). We also propose a qualitative evaluation in Figure 5.6, showing the parameters in UV space alongside the entropy maps (visualized using the "Turbo" colormap) for results optimized from images rendered under the "rural asphalt road" environment light. On the right columns we compare rerenderings of our results and Mitsuba's to ground-truth renderings under a different illumination, showing once more a good appearance match. We include visual comparisons on all synthetic cases as well as the *exr* files for all light environments in Supplemental Material.

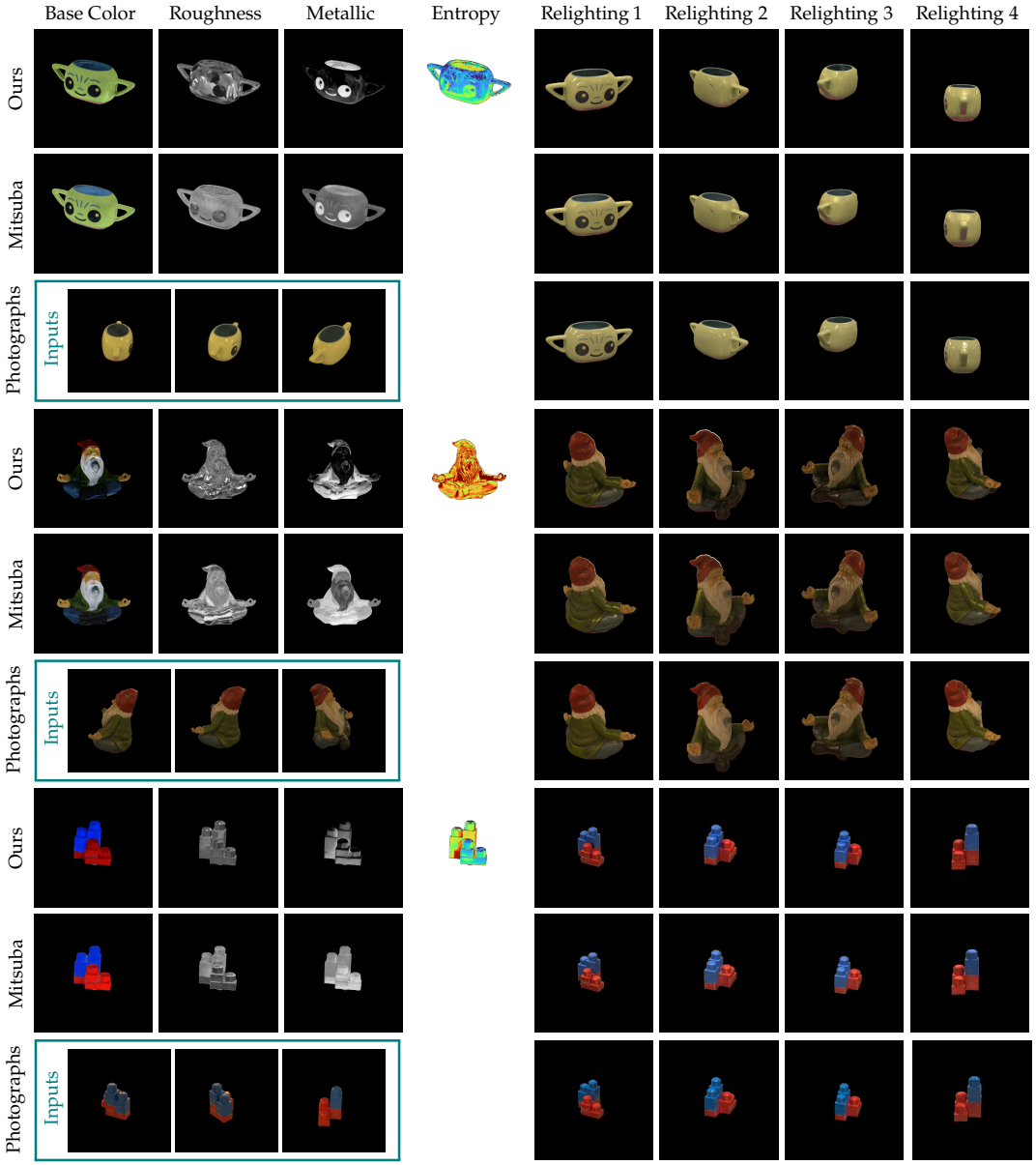


Figure 5.5: We compare our approach to Mitsuba and photographs for relighting. The first column on the left shows material parameters and training photographs, while the right column shows relit results and photographs under different lighting. We see that our results match the appearance of the photographs as well- or slightly better than Mitsuba. We see that the entropy is lower where inputs provide more information (e.g. well-lit regions). Videos in the Supplement.

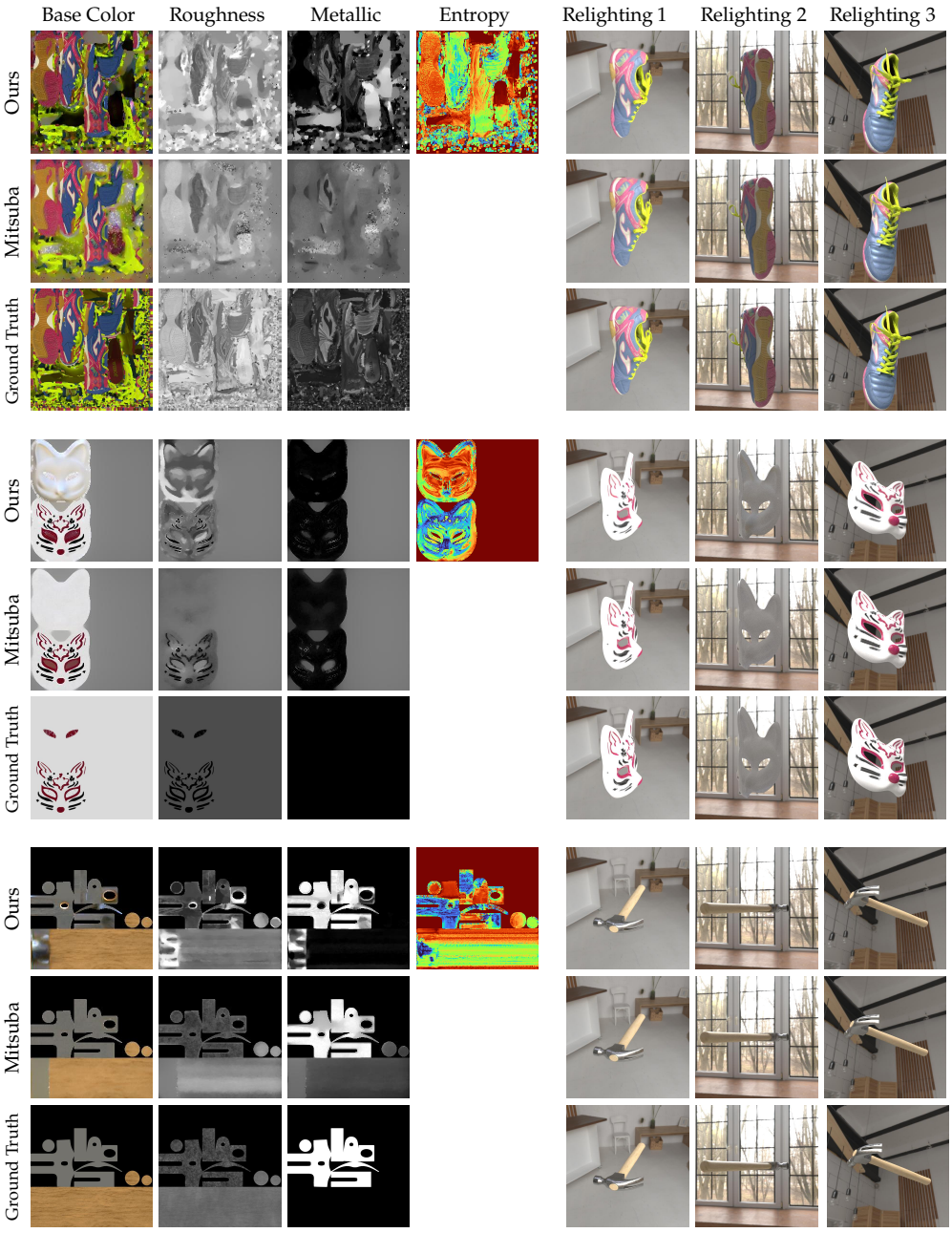


Figure 5.6: We compare our results to Mitsuba and ground truth parameters on synthetic data. We see that we achieve similar matching quality with a 35x speedup in the optimization. The relit renderings are from test views using a lighting environment unseen during optimization. Here too we see that entropy is higher where inputs provide less information (see supplemental materials for inputs).

Table 5.2: Synthetic benchmark comparison (MSE) of Mitsuba and our method to the ground-truth parameters. For each environment we present the error in the following way: Average (Base Color, Roughness, Metallicity). We also report for each environment the correlation between the entropy we compute with our power spectrum approximation and the error in Mitsuba's optimization to show that our entropy is useful for general inverse rendering pipelines.

	Photo Studio	Overcast	Museum	Rural Road	Avg.	Time
Mitsuba	0.04 (0.01, 0.03, 0.07)	0.03 (0.01, 0.02, 0.06)	0.04 (0.01, 0.04, 0.08)	0.04 (0.01, 0.03, 0.07)	0.04 (0.01, 0.03, 0.07)	105.19s
Ours	0.07 (0.02, 0.09, 0.10)	0.06 (0.02, 0.07, 0.09)	0.07 (0.02, 0.08, 0.11)	0.05 (0.01, 0.05, 0.08)	0.06 (0.02, 0.07, 0.09)	3.14s
Ours - Spectrum Only	0.19 (0.07, 0.15, 0.37)	0.20 (0.06, 0.14, 0.40)	0.25 (0.09, 0.20, 0.46)	0.22 (0.06, 0.16, 0.43)	0.21 (0.07, 0.16, 0.41)	1.20s
Error-Entropy correlation	0.21	0.13	0.18	0.14	0.16	-

5.5.4 Uncertainty

We test whether our proposed entropy-based uncertainty metric is indicative of error in inverse rendering results and representative for uncertainty in other inverse rendering frameworks.

Table 5.3: Comparing correlation between entropy computed with Mitsuba and our approximations.

Entropy using	Corr. Mitsuba ↑	Time
Mitsuba	1.00 ± 0.00	15m
Angular	0.90 ± 0.05	2.91s
Power Spectrum	0.89 ± 0.06	0.001s

Stanford ORB Our first claim is that the entropy computed using our power spectrum or other differentiable renderers will be similar. We validate this by computing our proposed entropy on the Stanford ORB dataset with Mitsuba, on a grid of $8 \times 8 \times 8$ parameters (roughness, metallicity, base color). We compute the Pearson correlation coefficient, ρ, between entropy computed with our power spectrum approximation and the one computed with Mitsuba's fully-fledged differentiable renderer. We find a high correlation between entropy computed with Mitsuba and both our mixed renderer $\rho = 0.90$ and the power spectrum approximation $\rho = 0.89$ as shown in Table 5.3. This validates that the entropy we compute with our power spectrum based model is indeed similar to that computed with a fully-fledged differentiable renderer. Further, as our power spectrum approximation is embarrassingly parallelizable, we obtain extreme speedups over both the mixed spherical harmonics method (2645×) and Mitsuba ($800,000\times$) making our uncertainty estimation very practical.

Synthetic dataset We also evaluate whether low entropy is associated with lower error after optimization, even for results optimized with another approach (Mitsuba), indicating if enough information is in the input. If entropy is high, we have a higher likelihood of incorrect parameters, though they might still be good. Indeed we do not expect a one-to-one correlation ($\rho = 1$). For example, if the likelihood for parameters in uncertain regions is uniform, low error in a high entropy part is equally likely as high error. That means a correlation of 0.5 is the most we can reasonably expect. We study this by computing the correlation between entropy computed using the power spectrum approximation and the squared error between the ground-truth material parameters and the parameters optimized with Mitsuba. On top of uncertainty, there could be other factors for error, which could result in lower correlation (such as

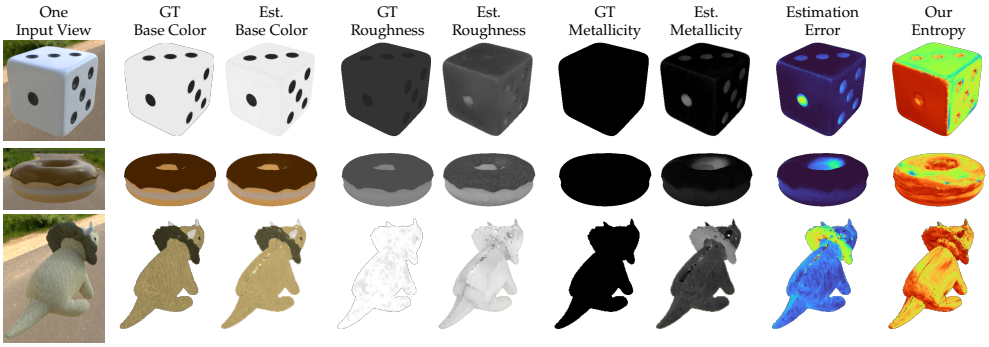


Figure 5.7: We demonstrate the informativeness of our fast approximate entropy for other inverse rendering frameworks (in this case, Mitsuba). We interleave ground truth properties and properties estimated with Mitsuba (base color, roughness, metallicity). On the right, we show the average estimation error followed by our entropy estimation, which is independent of Mitsuba's estimation. We observe that low entropy is indicative of lower error, suggesting that it captures the sufficiency of information in the input signal. On the dice example (top row), the 'one'-face is lit less than other faces and we observe highlights with lower intensity, leading to higher entropy. While we still recover the white albedo correctly, the estimation of roughness and metallicity for the dot has a high error. Similarly, the inside and lower parts of the doughnut are less observed and not lit by strong light sources. Again, entropy is high in regions of high error (especially in metallicity). The triceratops collar is down-facing and not well-lit and our entropy captures the lack of observation that leads to high error in the metallicity part.

global illumination approximations, approximate light environment, and incorrect geometry). In the last row of Table 5.2 we show the correlation scores. We show in Figure 5.7 results optimized from images rendered under the lighting environment "rural asphalt road" which presents a strong directional illumination (sun, outdoor, see supplemental folder). We can see that areas which are not directly lit by the sun in the input images exhibit higher entropy, intuitively, without enough observed specular signal, uncertainty is high (this is particularly visible on the dice example in the synthetic Supplemental Materials where 3 faces are well lit, and their opposed faces are not).

5.5.5 Ablations

Our ablations serve two goals. First, we want to validate that our proposed improvements to Ramamoorthi and Hanrahan [190] make a significant difference. Second, we would like to study the sensitivity of the approach to some hyperparameters (e.g. regularizer weight, number of spherical harmonics degrees). We vary our method and report the results on Stanford ORB and the synthetic dataset. Further ablations on spherical-harmonics fitting are in Appendix C.

BRDF model variants To understand whether our proposed improvements to the BRDF model by Ramamoorthi and Hanrahan [190] result in higher-quality acquisition results, we study variants of our model with and without the shadowing and masking terms. We run our ablations with a constant weight on all the samples. The results on Stanford ORB are presented in Table 5.4. We ob-

serve that the shadowing- and masking terms improve the results independently and we get the best results with both.

Convolutional only vs Directional One of the main benefits of our method is that we can study the effect of the BRDF fully in the spherical harmonics domain. We can perform the full BRDF recovery procedure in the spherical harmonic domain or the power spectrum domain, as done for our initialization and uncertainty. We show the impact of evaluating the BRDF parameters from the power spectrum in row four of Table 5.1, compared to running our our mixed frequency-directional optimization (third row). As these approximations do not account for effects such as Fresnel or shadowing/masking, we observe lower appearance matching, as expected. The approximations however still result in a very good appearance match and extremely fast computation. Most of the reported timings in Table 5.1 reflects the fitting of the spherical harmonics, which only has to be done once, letting us easily explore hundreds to thousands of possible parameter combinations in a fraction of a second to compute uncertainty.

5.5.6 Applications

In our applications, we show how fast acquisition and uncertainty can be used in (SV)BRDF capture to improve results, and guide understanding of error.

Initialization In the comparisons, our method demonstrated a fast and high-quality estimate of BRDF parameters. However, there is still benefit to finetuning our results with a differentiable path tracer. Such a step could finetune parts of the material that are affected by global illumination (e.g., in corners, or near highly reflective surfaces). In this experiment, we show the benefit of this approach by initializing material textures with the results from our method and optimizing them with Mitsuba for one epoch and only 16 samples per pixel (half the samples we used for the other experiments). This finetuning pass only costs 14s, for a total of 20s combined with out method as initialization. This is over three times faster than the 70s required without this initialization (Table 5.1). The results in Table 5.5 show that the combination of our approach and a finetuning pass with Mitsuba for one or two epochs achieves better

		Stanford ORB					Synthetic	
Shadowing	Masking	PSNR-H↑	PSNR-L↑	SSIM↑	LPIPS↓	Time	MSE	Time
-	-	25.646	32.809	0.973	0.035	1.09s	0.16	**2.57**s
✓	-	26.245	33.399	0.975	0.034	1.21s	0.14	2.81s
-	✓	26.070	33.101	0.974	0.035	1.19s	0.13	2.89s
✓	✓	**26.411**	**33.557**	**0.975**	**0.034**	1.31s	**0.12**	3.14s

Table 5.4: We ablate the different components of the BRDF. For this ablation we do not apply a cosine weighting to the samples, better highlighting the impact of the shadowing and masking terms.

Table 5.5: We apply our results as an initialization for a short fine-tuning pass using gradient descent with a differentiable path tracer (Mitsuba): we use only 1 epoch with 16 samples per pixel for the first three rows and 2 epochs with 32 samples per pixel for the fourth row. The final row shows the result of a full optimization for Mitsuba.

Initialization	PSNR-H↑	PSNR-L↑	SSIM↑	LPIPS↓	Total time
Constant	24.339	30.558	0.969	0.047	14.74s
Ours - Spectrum Only	25.745	33.158	0.975	0.033	16.29s
Ours	27.038	34.560	**0.978**	0.030	19.87s
Ours (2 epochs)	**27.199**	**34.770**	**0.978**	**0.029**	33.39s
Full optimization - Constant	26.595	34.185	0.977	0.032	69.96s

performance than only using Mitsuba (+0.5dB PSNR-H, PSNR-L) for less than a third of the time. We also surpass the results for only our method, which is to be expected.

Table 5.6: We combine the optimization results from multiple environments into one texture by using the parameter with the lowest certainty. We present the MSE in the following way: Average (Base Color, Roughness, Metallicity)

	MSE Synthetic
Average	0.04 (0.01, 0.03, 0.07)
arg min Entropy	0.03 (0.01, 0.02, 0.06)

Sharing information Points on the surface with low entropy have reasonable certainty and will likely exhibit lower error than parts with high entropy. We can use this information to select the best conditions to recover BRDF parameters, for example from different lighting setups. As a proof of concept, we merge the texture maps from the synthetic scenes based on entropy: for every texel, we use the parameters from the environment with the lowest entropy at that texel. The result in Table 5.6 shows that this simple approach beats the average over separate environments by 25%. The score is equal to the best-performing environment in Table 5.2. With this, we are able to select the best parameters without knowing the ground-truth. We believe that this approach would yield even stronger results in the context of more complementary environments.

Guiding capture Entropy can be applied during capture as a measure to define the most informative views. If a view decreases entropy, it's useful; if it does not, we could discard it. It is difficult to know what information a view will add without knowing its contents. A solution to this could be to compute an *expected* decrease in entropy (information gain) in case potential views are not known. Since this is not part of our core contribution, we leave this for future work. Instead, we show that entropy is useful to measure the informativeness of views. We ran the synthetic benchmark for one environment (Rural Road) and randomly drop out views for every optimization. In Figure 5.8, we show the corresponding average

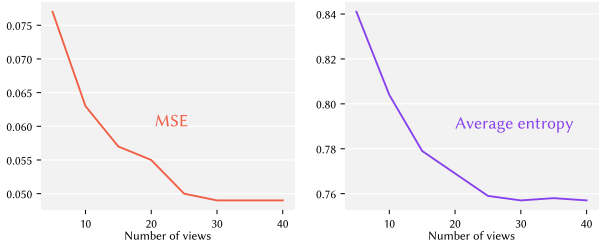


Figure 5.8: Plots of MSE and Entropy for varying input view counts on the synthetic benchmark with the Rural Road environment, showing that entropy can be used as a proxy of average MSE for a given set of views.

MSE over all shapes and channels and the average entropy over all shapes. We observe a clear relationship between the MSE and entropy, demonstrating that we can use entropy as a proxy for the expected success of an optimization. If one were to use entropy in real time, the alternatives to the power spectrum approximation quickly become tedious to use: for every frame that is considered, one would either have to wait a couple of seconds or minutes to compute the entropy (Table 5.3).

5.6 Challenges and conclusion

In summary, we present a material acquisition and uncertainty estimation method for multi-view capture of objects using a frequency domain analysis. We do so through efficient spherical harmonics fitting and a power spectrum approximation that lets us efficiently compute the error associated with varying material parameters for a surface. By interpreting this error as a likelihood function, we can use entropy as a measure of uncertainty. The results indicate correlation of low entropy with low error, both quantitatively and qualitatively. We show that entropy can be a useful proxy for acquisition quality. We also propose a way to take into account shadowing and masking to better reconstruct the target appearance and estimated properties, compared to the existing signal processing framework for inverse rendering. Our method yields results that are on par with the state-of-the-art for a 13x speedup in optimization and, combined with state-of-the-art, yields improved results for less than a third of the time.

We see the following challenges for future work. Our method requires HDR images as input, because the truncation applied for LDR images could introduce frequencies that are not present in the original signal. To make our method more accessible for use with LDR images, it is of interest to study how to use spherical harmonics decompositions on LDR images. This could be implemented, for example, as an amendment to the least-squares fitting procedure. Our current implementation only considers direct illumination, neglecting self-illumination and self-shadowing which may appear in challenging concave objects; our method could be extended to take them into account with a ray tracer, as our core theoretical and algorithmic contributions are defined for a general directional radiance field for points in space. Nonetheless, our method produces accurate estimations of materials and can be used as an initialization for more complete (and slower) differentiable renderer [102] for areas showing effects that we do not currently support.

We believe that our renewed exploration and improvement of the frequency-based model demonstrates the merits of this framework. We are excited about further applications that take advantage of entropy as a measure of uncertainty for material recovery.

6 Conclusion

This dissertation explored the use of intrinsic approaches to learning and computing on curved surfaces. We covered applications in machine learning, geometry processing, and appearance capture. The promise of using algorithms that work directly on curved surfaces in these applications is that they can simplify otherwise complex problems. This is because we observe information from inside the 2D surface, rather than their 3D extrinsic configuration. When this extrinsic information is not necessary for the task, it can be counterproductive, just like reading a message from a crumpled piece of paper is much harder than first flattening the paper. The challenge in each domain is twofold: first, we need to identify the underlying geometry and formulate the operations that would be beneficial for our task. Second, we have to deal with practical challenges from sparse and irregular sampling on curved domains.

For convolutions on surfaces, the intrinsic perspective lets us define convolution in 2D, rather than 3D, which is especially important for an expensive operation like convolution. The intrinsic perspective also makes our approaches robust to non-rigid deformations, which is necessary to generalize to real-world situations with many deformations. The challenge posed by the intrinsic perspective is that we have no global coordinate system, which complicates learning directional information, like edges and corners. We demonstrated fundamental solutions to this problem with rotation-equivariant kernels and parallel transport (HSN) and using coordinate-independent geometric operators from vector calculus (DeltaConv). We also had to deal with irregular and sparse sampling, which we approached with area weighting (HSN) and least-squares regularization (DeltaConv).

For geometric multigrid methods, the intrinsic approach offers a benefit over purely combinatoric approaches, such as Shi et al. [207], by providing faster convergence. Compared to extrinsic approaches in 3D, we can be more efficient. Also, the typical extrinsic approach would use mesh simplification, followed by a projection between hierarchy levels. This means that the hierarchies are not nested, which makes analysis difficult [116]. The challenge lies in creating a hierarchy and corresponding prolongation and restriction operators on a curved geometry, which can be slow to compute [140]. By using a carefully validated combination of geometric and combinatorial techniques with graph Voronoi diagrams, we were able to attain both fast convergence and hierarchy construction.

Finally, for material appearance capture, we observe that the incoming and outgoing light field is sampled on a curved domain:

the sphere. This insight leads to the formulation of reflection as a spherical convolution [190]. We show that this perspective enables faster SVBRDF capture and a quantitative analysis of uncertainty on the problem of SVBRDF recovery. The challenge of working directly on the sphere is again the sparse and irregular sampling of the radiance field. We demonstrate a robust and simple spherical harmonic transform using regularized least-squares that enables a high-speed analysis of uncertainty in the frequency domain.

6.1 Challenges and future work

A clear hurdle for using intrinsic approaches is that they require knowledge of the underlying geometry and specialized algorithms to deal with that geometry. A well known opinion piece by Richard Sutton, 'the bitter lesson', argues that the success of algorithms is determined by their scalability, not their ingenuity [222]. Algorithms that easily scale to large amounts of data and on parallelized hardware, Sutton argues, have historically won over more complicated algorithms. Following this line of reasoning, one could wonder whether the benefit of intrinsic approaches is worth the effort.

First, each of the algorithms presented is scalable in terms of data and parallelized execution. The main challenge is in scaling the *adoption* of these algorithms, which is a matter of ease-of-use and not a technical or theoretical limitation. The transition from rotation-equivariant kernels to geometric operators between HSN and DeltaConv was motivated by this consideration. We attempted to boil down the required operations to the simplest set of instructions using well-known and proven ideas. It proved difficult to demonstrate large improvement on the tasks that we chose to compare our approach on. Tasks that rely more on directional information and robustness to non-rigid deformation could demonstrate a more pronounced benefit. We also think that cases with limited data or compute resources can benefit from the geometric priors included by these methods.

We see a clear opportunity for scalability in combining the techniques developed for CNNs on surfaces (HSN and DeltaConv) and geometric multigrid. With Gravo Multigrid, we present a fast and simple hierarchy construction that could be used in conjunction with convolutional neural networks to enable learning on multiple grids. Such hierarchies have a proven track record for CNNs on images, for example, in U-Nets [194] and Vision Transformers [50]. We believe that a better application of hierarchies for 3D surfaces can help scale these techniques to larger inputs and enable learning of features at different levels of abstraction.

For our contributions on material capture, we demonstrate clear benefits of using an intrinsic (here, frequency domain) perspective: comparable performance in relighting with an optimization that is

an order of magnitude faster and new insights on uncertainty that would be multiple orders of magnitude slower with current state-of-the-art implementations. A challenge here lies in ensuring that each of the required inputs to our approach is available. At this moment, we require HDR photographs and a capture of the environment map. We believe that these inputs are feasible to recover, but a fully implemented pipeline that also accepts LDR photographs and recovers the environment lighting from the input photographs would make the adoption of our algorithms significantly simpler. We touched on applications of uncertainty estimation for SVBRDF recovery in chapter five and think these results are indicative of high potential for more applications of uncertainty estimation, for example: sharing data, guiding capture, and guiding the use of learned priors.

6.2 Closing words

We opened this dissertation with the history of Gauss and Riemann and the development of intrinsic analysis on curved surfaces. We applied these tools to learning and computing on curved surfaces, motivated by their effectiveness in mathematics and physics. The intrinsic perspective yields fundamental and conceptually simple solutions to otherwise complex problems. This is surely not the end of this story: there is ample room for application and extension of the presented approaches. However, borrowing words from Riemann, this leads us into the scope of future research, of which the object of this work does not allow us to go today.

7 Acknowledgements

The story of this dissertation probably started in a household store in Leiden. My mother brought me along to go shopping for groceries and a CRT-television showing clips of Toy Story 2 caught our eye. We bought two VHS tapes, one in Dutch and one in English. As I watched these tapes at home, the world of computer graphics and animation unfolded in front of me and made a deep impression. That day, and the numerous making-of documentaries I would later watch, launched my childhood dream to one day work at Pixar.

Fast forward to a lunch in 2014. I had just started as a student in computer science at the TU Delft and was taking the computer graphics course by Elmar Eisemann. We met to discuss possible projects for the honours track and I mentioned my interest in working on movies. Elmar responded that he had just recently been to Ed Catmull's (Pixar's founder) office on a visit for SIGGRAPH committee members. If I wanted to work there, I needed a PhD. In fact, some of his past collaborators were now working at Disney! There was a way in.

Ten years later, I am about to defend my dissertation. I am not working at Pixar, but I did walk through its Emeryville campus as seen in the making-of documentaries I watched as a child. I also learned that the place for inspiring, joyful, creative, and technically challenging work is not confined to a studio in California, but has its home in the field of computer graphics. I know, I can be sentimental. I want to thank Elmar for keeping my dreams alive, welcoming me, and guiding me in the field of computer graphics. Elmar exemplifies kindness, integrity, and curiosity and always knew where my biggest room for growth would be.

My first taste of computer graphics research was during my Master thesis, supervised by Klaus Hildebrandt. One day, we were walking along the Mekelpark, talking about topics for my thesis. I mentioned that I was interested in working on deep learning and computer graphics. If you want to do that, Klaus replied, you should consider geometric deep learning. So, my interest in computer graphics refined itself to the field of geometry processing. With Klaus' expert supervision, I graduated and thereafter continued to expand and publish the work at SIGGRAPH, which is now the first chapter in this dissertation. Klaus taught me to be thorough and precise. He would tell me to write only what I know and keep it to the point. I have maintained an appreciation for succinct writing ever since. Klaus is also an incredibly kind and thoughtful person. He would often come up to me to suggest that I should be looking out for this or that opportunity, or to start thinking about where I wanted to work next. I want to thank Klaus for mentoring me,

teaching by example, and helping me flourish.

After receiving my Master's degree, I continued as a PhD student. Joris Dik, my second promotor, and Elmar had joined funds to finance my position, for which I am incredibly grateful. Joris is not afraid to think big. In our first meeting, he spoke of publishing the work we would do in Nature and revolutionizing the field of art history. Both those goals turned out to be harder to achieve than I thought, especially during a global pandemic, but Joris' enthusiasm then and during our later meetings was invigorating.

I am grateful to have collaborated with many talented colleagues. With Dr. Ahmad Nasikun on DeltaConv and Gravo Multigrid. Nasikun is incredibly diligent, hard-working, humble, and kind. I have deep respect for how he balanced family life and doctoral work. With Yancong Lin, Jan van Gemert, and Silvia Pintea we worked on geometric priors for vanishing point detection. Yancong's grit and resilience in leading our paper to publication was inspiring, to say the least. Adrien Bousseau joined our lab for a sabbatical, in which we worked together on a (yet unpublished) project about unmixing paint. I have fond memories of our meetings, where Adrien was always his witty, sharp, sympathetic self. I would also like to thank Adrien for mentoring me along the way. I got to supervise Jules van der Toorn on his Bachelor thesis. Jules made an incredible effort on his thesis and even went on to publish his work at the 2022 Eurographics Workshop on Graphics and Cultural Heritage, winning the award for the best full paper.

In the field of art history, I had the great pleasure of working with Abbie Vandivere at the Mauritshuis, Francesca Gabrieli at the Rijksmuseum, Liselore Tissen at the TU Delft, and Sanne Frequin at Utrecht University. Abbie lead the effort on the "Who's that Girl" exhibition at the Mauritshuis, about the Girl with a Pearl Earring. Her genuine enthusiasm for art kept me motivated and I remember fondly the times nerding out over Bob Ross, scanning techniques, and the state of technology in art history. Francesca introduced me and my students to the use of hyperspectral data in painting analysis and welcomed us to the Rijksmuseum studio. Liselore and Sanne showed me how to make an impact in the physical world, 3D printing numerous paintings and shaping the story on 3D reproductions in the media.

In 2020, I met Valentin Deschaintre, while volunteering on the SIGGRAPH Research Career Development Committee, thanks to Justin Solomon, who facilitated and energized this community-building effort. In my third year, Valentin reached out to me and encouraged me to sign up for an internship at Adobe. "Be there, or be square!" This is typical of Valentin's light-hearted, but intentional style. His push got me out of my local bubble and gave me an opportunity to learn from some of the top researchers in the field. I am grateful to Valentin, Julien Philip, Miloš Hašan, Fujun Luan,

and Krishna Mullia at Adobe for making the internship a success. Also a big thank you to my colleagues and friends at Adobe who made my time in San Francisco truly special: Georgios, Mattia, Dani, Joanna, Yash, Crane, Julia, Arman, and Amir.

The work of a PhD student can be isolated, but is brightened by the companionship of lab mates. Many thanks to Alex, Mark, Annemieke, and Baran, for adventuring the open seas and the streets of oldtown and helping prepare and defend this dissertation. To Berend, for keeping my ego in check, while also providing companionship in fawning over famous professors. To Mathijs, for being my first deskmate and entertaining my themepark monologues. To Lukas and Jackson, for providing fresh energy to our group. To Nicolas, for the good coffee conversations. My heartfelt thanks to Soumyadeep, Jerry, Leo, Markus, Mijael, Guowei, Xuejiao, Ali, Faizan, Peiteng, Amir, Yang, Benno, and Mika, for the warmth they brought to the group. To the staff, Rafa, Thomas, Petr, Ricardo, Michael, and Martin for their wise perspectives and leadership. To Ruud and Lauretta, for always being ready to help with hardware and schedules. To Cynthia Liem, for her mentorship in the first years. My gratitude goes out to my colleagues abroad, Nicholas Sharp, Hsueh-Ti Derek Liu, Rana Hanocka, Olga Sorkine-Hornung, and Keenan Crane, for helping me figure out life in research.

Back to that place where my dreams were originally sparked, watching movies as a child. I am incredibly fortunate to have had such a joyful childhood in the context of a loving family. I want to thank my parents Haitse en Dientje, who encouraged me to step out and grow, while also loving me where I am. As my mother would say: just take the next step, you'll never know where you will end up. Thanks, to my siblings and their spouses, for supporting me and being a home where I could come to relax and re-energize: Foppe and Marieke, Jos and Suzanne, Eva and Arnold, Arjen and Anne-Marie, and Mirjam and Eeyore. And to my friends, who made sure I had a life outside of work. Finally, I dedicate this work to my dear Joanne. Thank you for sharing my attention with this work. Thank you for giving me a reason to finish my work on time, so I can be home with you. Thank you for listening, caring, debating, encouraging, and making me feel at home, wherever we go.

A Appendix DeltaConv

A.1 Discretized Operators

Gradient We construct a discrete gradient using the moving least-squares approach from [136]. We go through each step to show how to derive the gradient starting with the general formula from Riemannian geometry and simplify terms whenever the setting allows us to do so.

We locally fit a surface patch to estimate the metric at each point p using moving least-squares [167]. The surface patch $\mathbf{\Gamma} : \mathbf{\Omega} \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3$, often called a Monge patch, describes the surface as a quadratic polynomial $h(u, v)$ over the tangent plane at p and is given by

$$\mathbf{\Gamma}(u, v) = [u, v, h(u, v)]^\intercal , \tag{A.1}$$

where u, v denote local coordinates in the tangent plane. Since the surface patch should interpolate the point p and the surface normal of the patch at p should agree with the normal of the tangent plane at p, the constant and linear terms of $h(u, v)$ vanish

$$h(u, v) = \alpha_1 u^2 + \alpha_2 uv + \alpha_3 v^2, \tag{A.2}$$
$$h_u = 2\alpha_1 u + \alpha_2 v, \tag{A.3}$$
$$h_v = \alpha_2 u + 2\alpha_3 v. \tag{A.4}$$

The metric is given as

$$g = \begin{bmatrix} 1 + h_u^2 & h_u h_v \\ h_u h_v & 1 + h_v^2 \end{bmatrix}. \tag{A.5}$$

And its determinant as

$$|g| = (1 + h_u^2)(1 + h_v^2) - (h_u h_v)^2 \tag{A.6}$$
$$= 1 + h_u^2 + h_v^2 + h_u^2 h_v^2 - h_u^2 h_v^2 \tag{A.7}$$
$$= 1 + h_u^2 + h_v^2. \tag{A.8}$$

Finally, the inverse of g can be computed as

$$g^{-1} = \frac{1}{|g|} \begin{bmatrix} 1 + h_v^2 & -h_u h_v \\ -h_u h_v & 1 + h_u^2 \end{bmatrix}. \tag{A.9}$$

Conveniently, at the center point $h_u(0, 0) = h_v(0, 0) = 0$ and thus

$$g_{0,0} = g_{0,0}^{-1} = \mathbf{I}, \; |g_{0,0}| = 1. \tag{A.10}$$

To obtain nodes for the fitting of the quadratic polynomial, we

project the points from a local neighborhood of p onto the tangent plane. The gradient of a function X on the surface is given as

$$\operatorname{grad} X = \begin{bmatrix} \partial_u \mathbf{\Gamma} & \partial_v \mathbf{\Gamma} \end{bmatrix} g^{-1} \begin{bmatrix} \partial_u X \\ \partial_v X \end{bmatrix}, \tag{A.11}$$

where $\partial_u = \partial/\partial u$ is a shorthand for partial derivatives. Plugging Equation A.10 into Equation A.11, we get

$$\operatorname{grad} X = \partial_u X \partial_u \mathbf{\Gamma} + \partial_v X \partial_v \mathbf{\Gamma}. \tag{A.12}$$

$\partial_u \mathbf{\Gamma}$ and $\partial_v \mathbf{\Gamma}$ are exactly the basis vectors at point p. Thus, the coefficients of the resulting vectors are given by $\partial_u X$ and $\partial_v X$. The function X is given by function values at the points. To estimate the partial derivatives of X at a point p, we locally fit a quadratic polynomial using the same approach as for fitting a quadratic polynomial to the surface and compute its partial derivatives. As for the fitting of the surface patch, we project the points in a local neighborhood to the tangent plane and use the function values as nodes for fitting the quadratic polynomial [167].

Discrete Divergence The divergence, including the metric components [175], on the surface patch $\mathbf{\Gamma}$ is

$$\operatorname{div} V = \partial_u V_u + \partial_v V_v + V_u \partial_u \log \sqrt{|g|} + V_v \partial_v \log \sqrt{|g|}, \tag{A.13}$$

where $|g|$ denotes the determinant of the metric. At the origin, the metric of our surface patch is the identity and the derivatives of the metric at this point vanish. Hence, divergence is given by

$$\operatorname{div} V = \partial_u V_u + \partial_v V_v. \tag{A.14}$$

To compute the partial derivatives $\partial_u V_u, \partial_v V_v$ at p_i, we require the coefficients of the vector field at neighboring points $\{p_j \mid j \in \mathcal{N}_i\}$. However, different basis vectors are used at different points. Therefore, we need to map from the basis vectors at p_j to those of p_i. While doing so, we account for metric distortion by $\mathbf{\Gamma}$. The following equation requires a bit more notation to distinguish between vectors at different points. We denote the coordinates of p_j in the tangent space of p_i as (u_j, v_j), the metric of $\mathbf{\Gamma}$ at p_j as g_{u_j, v_j}, the coefficients of a tangent vector at p_j as (α_j^u, α_j^v), and the basis vectors at p_j as $\mathbf{e}_j^u, \mathbf{e}_j^v$. The coefficients of a vector at p_j in p_i's parameter domain are

$$g_{u_j, v_j}^{-1} \begin{bmatrix} \partial_u \mathbf{\Gamma}(u_j, v_j) \cdot \mathbf{e}_j^u & \partial_u \mathbf{\Gamma}(u_j, v_j) \cdot \mathbf{e}_j^v \\ \partial_v \mathbf{\Gamma}(u_j, v_j) \cdot \mathbf{e}_j^u & \partial_v \mathbf{\Gamma}(u_j, v_j) \cdot \mathbf{e}_j^v \end{bmatrix} \begin{bmatrix} \alpha_j^u \\ \alpha_j^v \end{bmatrix}. \tag{A.15}$$

Equation A.14 and Equation A.15 are combined to form a sparse matrix $\mathbf{D} \in \mathbb{R}^{N \times 2N}$ representing divergence.

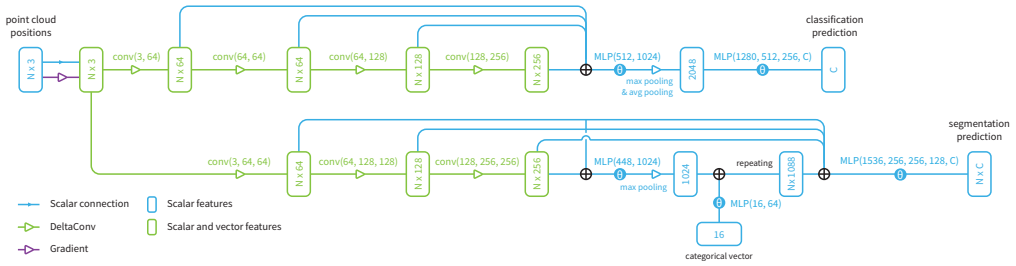


Figure A.1: The two architectures used for classification and segmentation, based on [236]. Please refer to Equation 6 and Figure 4 in the main text for the formulation of each convolution and how the streams are combined.

A.2 Architectures

We based our architectures on the designs proposed in DGCNN [236]. A schematic overview is presented in Figure A.1 and more details are provided in the following paragraphs.

Convolutions. Each convolution, denoted as $\textsc{Conv}(C_0, \dots, C_L)$, learns the function $h_{\boldsymbol{\Theta}}$ with an MLP that has L layers. Each layer in the MLP consists of a linear layer with C_i input- and C_{i+1} output channels, batch normalization [101], and a non-linearity. For scalar features, the non-linearity is a leaky ReLU with slope 0.2 and for vector features a ReLU. We denote MLPs applied per point as $\text{MLP}(C_0, \dots, C_L)$.

Classification network. The classification network has four convolution blocks: $\textsc{Conv}(3, 64)$, $\textsc{Conv}(64, 64)$, $\textsc{Conv}(64, 128)$, $\textsc{Conv}(128, 256)$. Each scalar convolution is interspersed with connections to- and from the vector stream, which mirrors the number of parameters in its vector convolutions. The output of each scalar convolution is concatenated into a feature vector of 512 features and transformed to 1024 features using an MLP. We return a global embedding by taking both the maximum and mean of the features over all points. These are concatenated and fed to a task-specific head: $\text{MLP}(2048, 512, 256, C)$, where C is the number of classes in the dataset. This final MLP has dropout [216] set to 0.5 in between the layers. During training, we optimize a smoothed cross-entropy loss.

Segmentation network. The segmentation network uses three convolutions: $\textsc{Conv}(C_{in}, 64, 64)$, $\textsc{Conv}(64, 128, 128)$, $\textsc{Conv}(128, 256, 256)$. Again, the scalar convolutions are interspersed with connections to- and from the vector stream. The output of each convolution is concatenated into a vector of 448 features per point and transformed to 1024 features with a global MLP. These features are pooled with maximum pooling. This embedding and an embedding of a one-hot encoding of the shape category is concatenated to the output of the convolutions at each point and fed to the task-specific head for segmentation: $\text{MLP}(1536, 256, 256, 128, C)$. During training, we optimize a cross-entropy loss.

U-ResNet architecture The U-ResNet architecture follows the design proposed in KPFCNN [224] (Figure 9 of the supplementary material in [224]). This network consists of an encoder that operates on four scales and a decoder that progressively upsamples the features to the original resolution. In each scale of the encoder, there are two ResNet blocks with a bottleneck. In KPFCNN, the first ResNet block uses strided convolutions, which we replace with pooling followed by a regular ResNet block. Each scale, we subsample to 1/4 points and increase the number of features by two. In the first layer, we use 64 features. We add two additional ResNet blocks with 128 output features after the decoder, as this was shown to be beneficial in CurveNet [254]. We do not use the other changes introduced by CurveNet, such as skip attention in the decoder or squeeze-excitation in the task-specific head. Each convolution block is replaced by a DeltaConv block, which maintains a vector stream in the first three scales and in the final two ResNet blocks. During pooling, scalar features are max-pooled and vector features are averaged with parallel transport to the coordinate system of the sampled point [244].

A.3 Additional Results and Visualizations

A.3.1 ShapeNet

The per-category breakdown of results for ShapeNet are listed in Table A.1.

A.3.2 Visualizations

The anisotropic diffusion experiment was repeated for another input image with 20 anisotropic diffusion steps (Figure A.2) and with varying anisotropic diffusion steps (Figure A.3), showing that a DeltaConv network can approximate anisotropic diffusion for varying diffusion times.

Table A.1: Per-category breakdown of part segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points.

	Mean inst. mIoU	aero	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
# shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
PointNet++	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
PointCNN	86.1	84.1	86.5	86.0	80.8	90.6	79.7	92.3	88.4	85.3	96.1	77.2	95.3	84.2	64.2	80.0	83.0
DGCNN	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
KPConv deform	86.4	84.6	86.3	87.2	81.1	91.1	77.8	**92.6**	88.4	82.7	96.2	**78.1**	95.8	85.4	**69.0**	**82.0**	83.6
KPConv rigid	86.2	83.8	86.1	88.2	**81.6**	91.0	80.1	92.1	87.8	82.2	96.2	77.9	95.7	**86.8**	65.3	81.7	83.6
GDANet	86.5	84.2	88.0	**90.6**	80.2	90.7	**82.0**	91.9	88.5	82.7	96.1	75.8	95.7	83.9	62.9	83.1	**84.4**
PointTransformer	86.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PointVoxelTransformer	86.5	85.1	82.8	88.3	81.5	**92.2**	72.5	91.0	88.9	85.6	95.4	76.2	94.7	84.2	65.0	75.3	81.7
CurveNet	86.8	85.1	84.1	89.4	80.8	91.9	75.2	91.8	88.7	**86.3**	96.3	72.8	95.4	82.7	59.8	78.5	84.1
DeltaNet (ours)	86.6	84.9	82.8	89.1	81.3	91.9	79.7	92.2	88.6	85.5	**96.7**	77.2	**95.8**	83.0	61.1	77.5	83.1
Delta-U-ResNet (ours)	**86.9**	**85.3**	**88.1**	88.6	81.4	91.8	78.4	92.0	**89.3**	85.6	96.1	76.4	**95.9**	82.7	65.0	76.6	84.1

Figure A.2: Comparison of different convolutions optimized to match the result of twenty anisotropic diffusion steps on sample image 'camera'.

Figure A.3: Comparison of a ResNet with DeltaConv optimized to match the result of varying anisotropic diffusion steps.

Table A.2: Results on human part segmentation [152].

Method	Accuracy
PointNet++ [187]	90.8
MDGCNN [184]	88.6
DGCNN [236]	89.7
SNGC [84]	91.0
HSN [244]	91.1
MeshWalker [121]	**92.7**
CGConv [262]	89.9
FC [162]	92.5
DiffusionNet - xyz [201]	90.6
DiffusionNet - hks [201]	91.7
DeltaNet - xyz	92.2

A.3.3 Human Shape Segmentation

We trained a variant of the simple single-scale DeltaNet (eight layers with each 128 channels) to predict part annotations on the human body dataset proposed by Maron et al. [152]. This training set is composed of meshes from FAUST (100 shapes) [12], SCAPE (71 shapes) [7], Adobe Mixamo (41 shapes) [1], and MIT (169 shapes) [232]. SHREC07 (18 shapes) is used for testing. Each dataset contains human bodies in different styles and poses, e.g., realistic, cartoony, dynamic. We convert the dataset into a point cloud dataset by uniformly sampling $8N$ points from the faces and downsampling these to N points with FPS. We set $N = 1024$, similar to the experiments in Wiersma et al. [244], $k = 20$ and $\lambda = 0.001$, similar to the other experiments. We normalize the area of the shape before sampling points and augment the input to the network with random rotations around the up-direction, a random scale between 0.8 and 1.25, and a random translation of 0.1 points. The network is optimized with Adam [111] for 50 epochs with an initial learning rate of 0.01. The results are listed in Table A.2. This experiment shows DeltaConv's effectiveness on a deformable shape class and allows us to compare the results to those of other intrinsic (mesh) convolutions. This comparison has its limits, as most of the listed methods are trained on meshes instead of point clouds. Nonetheless, we find that DeltaConv is in line with state-of-the-art approaches, with only raw xyz coordinates as input.

Appendix GravoMG B

Table B.1 shows more results for the Poisson problem on manifold meshes. We report a comparison on data smoothing with $\alpha = 1 \times 10^{-3}$ for manifold meshes in Table B.2 and non-manifold meshes and point clouds in Table B.3. Convergence plots on data smoothing on the manifold meshes in the main paper are shown in Figure B.1 and Figure B.2.

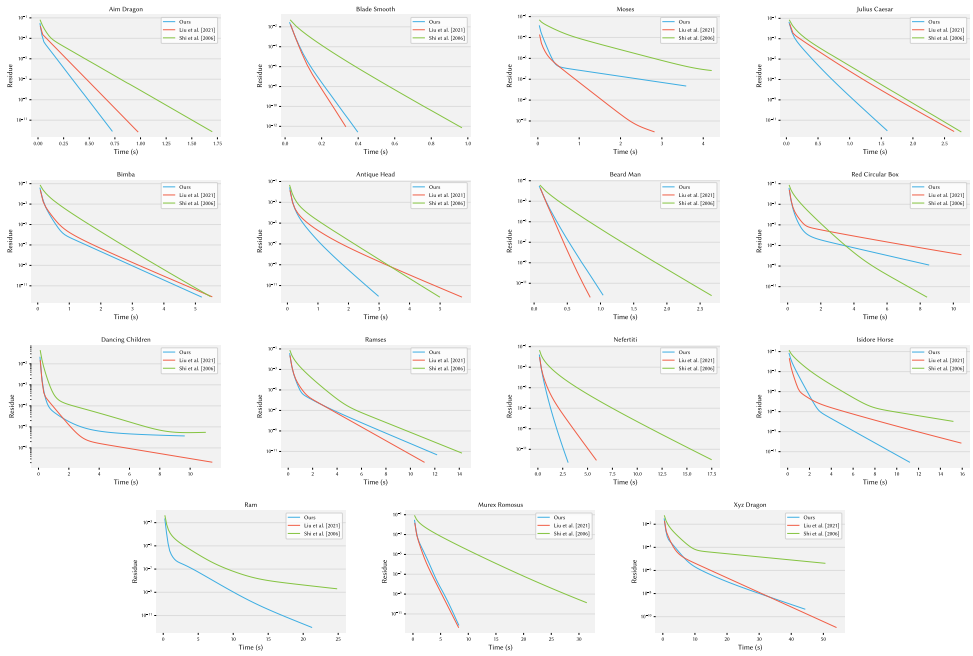


Figure B.1: Convergence plots showing **time** on the x-axis for smoothing with $\alpha = 1 \times 10^{-3}$.

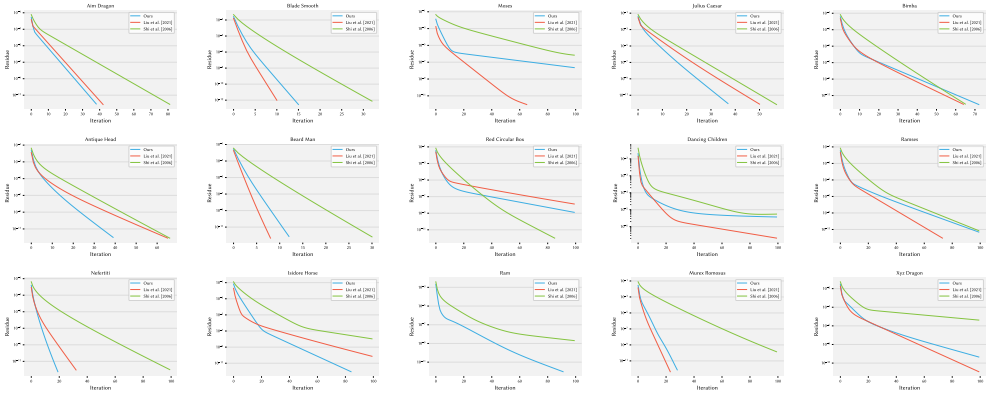


Figure B.2: Convergence plots showing **iterations** on the x-axis for smoothing with $\alpha = 1 \times 10^{-3}$.

Table B.1: Comparison of our hierarchy construction and solver for a **Poisson problem** with $\eta = 1 \times 10^{-6}$ mass matrix coefficient and tolerance of 1×10^{-4}. The maximum number of iterations for iterative solvers is set to 100.

		Gravo MG (Ours)			Liu et al.			Shi et al.			AMG-RS			AMG-SA			Eigen		Pardiso	
Model	#Vert	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Fact.	Subst.	Fact.	Subst.
Beetle	19k	0.01	28	0.06	0.55	18	0.05	0.01	33	0.09	0.03	100	0.29	0.03	100	0.20	0.02	0.00	0.06	0.00
Ogre	19k	0.02	6	0.02	0.54	7	0.03	0.01	13	0.04	0.03	100	0.27	0.04	40	0.08	0.02	0.00	0.06	0.00
Screwdriver	27k	0.02	4	0.03	0.75	4	0.03	0.01	9	0.04	0.03	100	0.35	0.05	24	0.07	0.05	0.00	0.09	0.00
Mumble	34k	0.03	10	0.05	0.97	6	0.04	0.02	9	0.04	0.04	100	0.41	0.05	57	0.17	0.04	0.00	0.11	0.01
Horse	48k	0.04	7	0.05	1.42	6	0.06	0.02	12	0.08	0.05	100	0.58	0.08	41	0.19	0.10	0.00	0.14	0.01
Laurent's Hand	50k	0.04	5	0.07	1.59	4	0.06	0.02	23	0.18	0.06	100	0.66	0.10	36	0.21	0.10	0.00	0.18	0.01
Dinosaur	56k	0.04	7	0.06	1.71	5	0.06	0.02	17	0.14	0.06	100	0.65	0.09	49	0.28	0.08	0.00	0.18	0.01
Heart	78k	0.06	15	0.18	2.90	29	0.38	0.04	33	0.43	0.08	100	1.06	0.13	100	0.95	0.17	0.01	0.27	0.01
Hannya Mask	83k	0.07	15	0.19	2.56	4	0.09	0.05	19	0.26	0.09	100	1.03	0.13	100	0.91	0.17	0.01	0.30	0.02
Trex	100k	0.10	9	0.16	3.40	4	0.14	0.05	42	0.64	0.12	100	1.39	0.22	97	1.10	0.18	0.01	0.38	0.02
The Thinker	110k	0.09	4	0.08	3.50	4	0.11	0.04	13	0.19	0.10	100	1.10	0.20	26	0.26	0.36	0.01	0.39	0.02
Egea	134k	0.11	4	0.13	4.42	4	0.14	0.06	15	0.31	0.16	100	1.79	0.23	27	0.36	0.94	0.02	0.50	0.03
Sappho's Head	140k	0.11	6	0.17	4.36	7	0.21	0.07	14	0.35	0.15	100	1.79	0.24	45	0.73	0.38	0.01	0.51	0.03
Human Torso	142k	0.15	4	0.15	4.97	5	0.20	0.06	40	0.85	0.16	100	1.86	0.25	44	0.72	0.73	0.02	0.57	0.03
Aim Dragon	152k	0.14	7	0.19	5.14	8	0.27	0.06	27	0.62	0.15	26	0.46	0.31	29	0.48	0.81	0.02	0.58	0.04
Armadillo	172k	0.16	9	0.28	5.97	7	0.29	0.08	27	0.76	0.20	100	2.37	0.36	52	1.04	0.54	0.02	0.62	0.04
Ronaldo	176k	0.18	8	0.28	6.32	6	0.30	0.10	40	1.19	0.21	100	2.74	0.40	75	1.72	0.60	0.02	0.69	0.04
Isis	187k	0.16	4	0.18	6.25	3	0.17	0.09	15	0.45	0.22	100	2.66	0.31	70	1.39	1.12	0.02	0.65	0.04
Blade Smooth	195k	0.15	4	0.17	5.87	3	0.17	0.09	18	0.60	0.18	100	2.42	0.40	42	0.94	0.76	0.02	0.69	0.04
Max Planck	199k	0.16	7	0.26	5.98	4	0.19	0.08	19	0.55	0.17	100	2.20	0.30	34	0.69	1.94	0.03	0.67	0.05
Vase-Lion	200k	0.17	4	0.20	6.54	4	0.22	0.12	10	0.43	0.21	100	2.68	0.33	36	0.88	0.40	0.02	0.79	0.04
Duck	204k	0.16	8	0.26	5.91	7	0.27	0.08	16	0.45	0.18	100	2.47	0.37	69	1.39	1.68	0.03	0.64	0.05
Mouse	214k	0.19	6	0.23	6.33	4	0.20	0.10	21	0.61	0.20	100	2.60	0.39	60	1.28	1.65	0.03	0.70	0.05
Wolf Skull	228k	0.23	7	0.33	8.07	5	0.33	0.13	38	1.52	0.27	100	3.47	0.53	53	1.59	0.39	0.02	0.92	0.05
Moses	258k	0.42	12	0.55	8.72	5	0.35	0.30	100	4.30	0.27	100	3.64	0.55	100	3.33	0.90	0.02	1.04	0.05
Rockerarm	271k	0.27	3	0.20	9.20	3	0.27	0.16	5	0.34	0.28	100	2.90	0.52	22	0.53	1.79	0.03	1.01	0.06
Pulley2	293k	0.24	5	0.32	8.87	4	0.30	0.14	24	1.04	0.28	100	3.67	0.58	38	1.19	3.03	0.04	1.07	0.07
Heraklion	350k	0.36	20	1.18	12.95	7	0.67	0.21	100	5.79	0.46	100	5.82	0.80	100	4.54	2.39	0.04	1.45	0.08
Julius Caesar	387k	0.30	11	0.58	12.10	17	1.09	0.16	28	1.54	0.39	100	4.89	0.77	70	2.79	5.07	0.06	1.29	0.09
Goyle	393k	0.33	2	0.21	11.44	2	0.26	0.21	12	0.73	0.39	100	4.75	0.79	25	0.97	8.43	0.07	1.32	0.10
Eros	476k	0.41	NaN	NaN	14.78	5	0.55	0.21	28	1.90	0.46	100	6.01	0.97	71	3.55	4.03	0.06	1.76	0.11
Roal	484k	0.47	5	0.54	16.52	4	0.59	0.37	37	3.27	0.53	100	6.86	0.84	53	3.08	1.34	0.04	2.08	0.09
Skeleton	494k	0.71	9	1.10	20.60	NaN	NaN	0.35	100	9.18	0.66	100	9.42	1.63	100	7.82	3.33	0.05	2.33	0.11
Bimba	502k	0.43	7	0.65	15.58	6	0.74	0.24	48	4.16	0.51	100	6.97	1.15	69	4.45	3.54	0.05	2.13	0.10
Oil Pump	570k	0.47	19	1.57	18.76	12	1.31	0.25	45	3.93	0.55	100	7.29	1.27	74	4.92	6.85	0.07	2.30	0.13
Antique Head	651k	0.53	4	0.51	20.07	4	0.60	0.28	14	1.22	0.64	100	8.48	1.37	67	4.33	15.02	0.11	2.43	0.17
Pulley	660k	0.54	19	1.81	21.54	16	1.88	0.28	53	5.23	0.63	100	8.46	1.49	58	4.31	12.94	0.10	2.68	0.16
Beard Man	691k	0.59	4	0.56	22.15	3	0.58	0.26	14	1.43	0.52	100	7.07	1.46	14	0.96	24.57	0.14	2.72	0.19
Red Circular Box	701k	0.64	6	0.74	22.97	6	0.97	0.34	66	6.67	0.72	100	8.98	1.51	66	5.01	17.76	0.11	2.84	0.17
Dancing Children	724k	0.69	9	1.10	23.21	8	1.25	0.41	39	4.54	0.68	100	9.32	1.23	100	8.70	6.18	0.09	2.78	0.17
John The Baptist	750k	0.73	69	7.74	29.90	10	1.76	0.47	81	10.56	0.86	100	10.53	1.86	100	9.77	6.43	0.08	3.08	0.17
Ramses	826k	0.79	7	1.21	28.90	5	1.18	0.47	40	6.03	0.91	100	11.82	2.08	49	5.40	6.24	0.09	3.60	0.18
Nicolo Da Uzzano	946k	0.76	7	1.02	29.88	6	1.16	0.40	33	4.15	0.85	100	12.08	2.05	82	8.06	17.75	0.16	3.53	0.24
Raptor	1m	0.89	NaN	NaN	32.27	NaN	NaN	0.43	NaN	NaN	1.07	100	13.98	1.83	100	12.14	0.79	0.00	3.77	0.20
Nefertiti	1m	0.89	4	0.94	34.22	4	1.18	0.56	65	11.69	1.09	100	14.74	2.58	46	6.14	9.15	0.10	4.54	0.22
Isidore Horse	1.1m	1.12	11	1.90	35.60	5	1.27	0.50	70	11.03	1.11	100	14.60	2.50	88	10.72	24.01	0.17	4.56	0.28
Horse Head	1.3m	2.42	44	10.76	54.11	10	4.52	1.00	100	24.52	2.12	100	24.57	4.25	100	20.97	12.99	0.13	6.55	0.30
Ram	1.3m	2.40	3	1.95	56.18	NaN	NaN	1.14	49	13.39	2.45	100	25.95	4.72	100	21.71	17.07	0.15	6.99	0.33
Murex Romosus	1.8m	2.32	6	2.85	73.05	5	3.32	0.99	63	20.79	2.38	100	29.83	5.08	58	15.28	40.06	0.26	9.13	0.44
XYZ Dragon	3.6m	3.24	9	5.32	121.97	7	5.32	1.57	55	28.95	3.16	100	43.75	9.14	75	30.54	77.62	0.69	15.88	0.94

Table B.2: Comparison of our hierarchy construction and solver for **data smoothing** of a random function with smoothing coefficient $\alpha = 1 \times 10^{-3}$ and tolerance of 1×10^{-4}. The maximum number of iterations for iterative solvers is set to 100.

		Gravo MG (Ours)			Liu et al.			Shi et al.			AMG-RS			AMG-SA			Eigen		Pardiso	
Model	#Vert	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Fact.	Subst.	Fact.	Subst.
Beetle	19k	0.01	27	0.06	0.55	12	0.04	0.01	24	0.07	0.03	75	0.22	0.04	92	0.18	0.02	0.00	0.06	0.00
Ogre	19k	0.01	6	0.02	0.54	7	0.03	0.01	8	0.02	0.03	12	0.03	0.04	16	0.03	0.02	0.00	0.06	0.00
Screwdriver	27k	0.02	4	0.03	0.75	4	0.03	0.01	7	0.03	0.03	14	0.05	0.04	12	0.03	0.05	0.00	0.08	0.00
Mumble	34k	0.03	9	0.04	1.00	6	0.04	0.02	7	0.04	0.04	29	0.12	0.05	46	0.13	0.04	0.00	0.11	0.01
Horse	48k	0.04	6	0.04	1.38	6	0.06	0.02	8	0.06	0.05	34	0.20	0.07	15	0.07	0.10	0.00	0.14	0.01
Laurent's Hand	50k	0.04	5	0.07	1.56	4	0.06	0.02	9	0.08	0.06	17	0.11	0.11	12	0.07	0.10	0.00	0.17	0.01
Dinosaur	56k	0.04	5	0.05	1.72	4	0.06	0.02	7	0.07	0.06	57	0.37	0.09	22	0.13	0.08	0.00	0.18	0.01
Heart	78k	0.06	16	0.19	2.92	26	0.34	0.04	16	0.22	0.08	7	0.08	0.13	60	0.58	0.17	0.01	0.27	0.01
Hannya Mask	83k	0.07	8	0.11	2.56	3	0.08	0.05	11	0.16	0.08	34	0.35	0.13	100	0.92	0.17	0.01	0.30	0.02
Trex	100k	0.10	7	0.13	3.40	4	0.14	0.04	12	0.21	0.13	100	1.40	0.18	23	0.26	0.20	0.01	0.38	0.02
The Thinker	110k	0.09	4	0.08	3.48	3	0.10	0.04	7	0.11	0.10	20	0.22	0.21	11	0.12	0.35	0.01	0.39	0.02
Egea	134k	0.11	5	0.14	4.40	4	0.14	0.06	7	0.17	0.15	35	0.63	0.22	12	0.16	0.90	0.02	0.50	0.03
Sappho's Head	140k	0.11	6	0.16	4.38	7	0.22	0.07	8	0.22	0.14	100	1.79	0.29	16	0.26	0.38	0.01	0.51	0.03
Human Torso	142k	0.15	4	0.15	4.88	4	0.17	0.06	10	0.24	0.16	97	1.77	0.25	12	0.19	0.74	0.02	0.57	0.04
Aim Dragon	152k	0.13	6	0.17	5.07	8	0.27	0.06	14	0.35	0.15	6	0.11	0.31	10	0.17	0.80	0.02	0.58	0.04
Armadillo	172k	0.15	9	0.27	5.88	7	0.28	0.08	20	0.57	0.20	15	0.36	0.35	34	0.68	0.54	0.02	0.62	0.04
Ronaldo	176k	0.18	7	0.26	6.35	5	0.26	0.10	13	0.44	0.20	64	1.75	0.40	38	0.88	0.62	0.02	0.69	0.04
Isis	187k	0.16	3	0.16	6.29	3	0.18	0.09	10	0.32	0.22	23	0.61	0.31	46	0.92	1.13	0.02	0.65	0.04
Blade Smooth	195k	0.15	4	0.17	5.85	3	0.17	0.09	8	0.30	0.18	21	0.51	0.40	16	0.36	0.76	0.02	0.69	0.04
Max Planck	199k	0.16	7	0.26	5.95	4	0.20	0.08	8	0.26	0.18	100	2.20	0.30	20	0.42	1.97	0.03	0.67	0.05
Vase-Lion	200k	0.17	3	0.17	6.34	2	0.15	0.12	3	0.17	0.20	12	0.32	0.41	11	0.27	0.37	0.02	0.76	0.04
Duck	204k	0.15	6	0.21	5.89	5	0.21	0.08	11	0.32	0.19	100	2.46	0.37	25	0.50	1.66	0.03	0.65	0.05
Mouse	214k	0.19	5	0.21	6.38	4	0.20	0.10	11	0.35	0.20	100	2.59	0.40	31	0.66	1.68	0.03	0.71	0.05
Wolf Skull	228k	0.23	5	0.26	8.01	4	0.29	0.13	13	0.57	0.26	44	1.50	0.51	32	0.94	0.38	0.02	0.91	0.04
Moses	258k	0.48	7	0.37	8.64	4	0.31	0.36	44	1.95	0.27	100	3.64	0.45	100	3.34	0.92	0.02	1.04	0.05
Rockerarm	271k	0.27	3	0.21	9.21	3	0.27	0.17	3	0.27	0.28	22	0.64	0.53	10	0.25	1.84	0.03	1.01	0.06
Pulley2	293k	0.24	4	0.29	8.81	4	0.29	0.14	10	0.49	0.28	96	3.49	0.69	23	0.72	3.07	0.04	1.07	0.07
Heraklion	350k	0.36	13	0.84	12.83	5	0.55	0.20	24	1.52	0.46	100	5.85	0.80	100	4.65	2.43	0.04	1.45	0.08
Julius Caesar	387k	0.30	7	0.41	11.61	9	0.63	0.16	12	0.68	0.36	100	4.89	0.75	36	1.44	5.22	0.06	1.28	0.09
Goyle	393k	0.32	2	0.20	11.39	2	0.26	0.21	4	0.32	0.37	14	0.67	0.79	11	0.43	8.43	0.07	1.32	0.10
Eros	476k	0.41	7	0.55	14.80	5	0.55	0.21	11	0.84	0.46	100	5.98	1.19	35	1.73	4.18	0.07	1.77	0.12
Roal	484k	0.47	4	0.47	16.50	4	0.59	0.38	12	1.20	0.53	64	4.40	1.08	19	1.10	1.31	0.04	2.08	0.09
Skeleton	494k	0.69	7	0.95	20.03	NaN	NaN	0.34	18	1.95	0.69	100	9.45	1.65	41	3.23	3.45	0.05	2.32	0.11
Bimba	502k	0.43	6	0.58	15.73	5	0.65	0.24	14	1.34	0.52	100	7.05	0.89	35	2.28	3.54	0.05	2.13	0.10
Oil Pump	570k	0.47	18	1.51	18.79	11	1.22	0.25	30	2.67	0.55	90	6.62	1.30	46	3.10	7.13	0.07	2.36	0.13
Antique Head	651k	0.53	4	0.51	20.10	4	0.60	0.28	9	0.84	0.64	7	0.60	1.37	44	2.82	14.65	0.11	2.42	0.18
Pulley	660k	0.54	18	1.70	21.62	14	1.66	0.28	27	2.77	0.62	100	8.47	1.48	46	3.43	12.80	0.10	2.68	0.16
Beard Man	691k	0.58	4	0.56	22.44	3	0.58	0.26	7	0.81	0.52	5	0.36	1.78	6	0.42	24.65	0.14	2.71	0.19
Red Circular Box	701k	0.64	6	0.74	22.89	6	0.96	0.33	15	1.69	0.72	100	8.94	1.51	44	3.32	17.84	0.11	2.83	0.17
Dancing Children	724k	0.69	7	0.89	23.09	9	1.35	0.40	21	2.54	0.68	100	9.27	1.97	58	4.98	6.34	0.09	2.77	0.17
John The Baptist	750k	0.72	14	1.78	29.71	6	1.19	0.47	14	2.09	0.85	100	10.45	1.78	47	4.59	6.16	0.08	3.06	0.17
Ramses	826k	0.80	5	0.96	29.00	5	1.19	0.47	15	2.46	0.90	100	11.58	2.10	22	2.39	5.97	0.08	3.58	0.18
Nicolo Da Uzzano	946k	0.77	7	1.02	30.04	5	1.02	0.46	12	1.66	0.85	100	12.03	2.01	33	3.24	19.50	0.16	3.65	0.25
Raptor	1m	0.90	36	5.22	32.22	NaN	NaN	0.43	52	8.35	1.07	100	13.97	2.40	92	11.14	4.61	0.11	4.24	0.23
Nefertiti	1m	0.89	4	0.94	34.34	4	1.18	0.57	14	2.79	1.09	57	8.41	2.55	14	1.87	9.23	0.11	4.56	0.22
Isidore Horse	1.1m	0.94	10	1.65	35.42	5	1.28	0.50	24	3.98	1.10	100	14.46	2.47	74	8.89	23.25	0.17	4.54	0.28
Horse Head	1.3m	2.36	20	5.71	53.36	11	4.80	1.00	74	19.51	2.18	100	26.10	4.43	100	22.69	12.88	0.14	6.55	0.31
Ram	1.3m	2.72	3	2.18	57.16	NaN	NaN	1.21	5	2.53	2.32	83	21.68	4.68	25	5.49	17.59	0.15	7.05	0.33
Murex Romosus	1.8m	2.30	5	2.56	72.19	4	2.87	1.01	26	8.99	2.32	100	29.59	5.16	24	6.30	38.34	0.25	9.06	0.43
XYZ Dragon	3.6m	3.30	9	5.35	123.64	7	5.29	1.56	18	10.27	3.25	100	43.72	8.84	67	27.36	79.75	0.74	16.09	0.96

Table B.3: Comparison of our hierarchy construction and solver for **data smoothing** of a random function with smoothing coefficient $\alpha = 1 \times 10^{-3}$ and tolerance of 1×10^{-4} on non-manifold meshes and point clouds. The maximum number of iterations for iterative solvers is set to 100.

		Gravo MG (Ours)			Shi et al.			AMG-RS			AMG-SA			Eigen		Pardiso	
Model	#Vert	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Hier	#It	Solve	Fact.	Subst.	Fact.	Subst.
NON-MANIFOLD TRIANGULAR MESHES																	
Lakoon	188k	0.16	5	0.20	0.09	26	0.88	0.20	100	2.76	0.40	17	0.39	0.43	0.02	0.71	0.04
Indonesian Statue	294k	0.26	7	0.42	0.17	7	0.44	0.30	37	1.45	0.63	100	3.53	0.90	0.03	1.18	0.06
Beethoven	383k	0.45	3	0.42	0.22	6	0.56	0.52	5	0.33	0.94	14	0.75	2.45	0.04	1.66	0.09
Bayon Lion	749k	1.44	4	1.20	0.71	8	1.67	1.37	7	1.07	2.45	10	1.28	6.26	0.08	3.75	0.18
Helmet Moustache	941k	2.04	5	2.07	0.74	22	4.97	2.10	15	2.97	3.44	17	2.79	24.99	0.14	5.56	0.26
Zeus	1.3m	2.51	7	2.69	1.20	14	4.52	2.49	29	7.24	4.19	100	20.66	30.96	0.19	7.26	0.35
Alfred Jacquemart	1.4m	3.26	4	3.33	1.68	13	6.16	3.61	7	2.44	5.24	28	8.24	9.08	0.14	8.03	0.35
POINT CLOUDS																	
Oil Pump	103k	0.07	4	0.07	0.04	6	0.11	0.10	7	0.09	0.19	12	0.13	0.17	0.01	0.30	0.02
Caesar Merged	388k	0.29	4	0.30	0.17	7	0.52	0.41	7	0.39	0.83	31	1.40	4.81	0.06	1.50	0.10
Truck	1.2m	0.96	6	1.39	0.68	9	2.14	1.27	12	2.30	3.69	46	6.88	5.81	0.15	5.24	0.29
Ignatius	1.4m	1.25	6	1.67	0.78	15	4.14	1.58	30	6.44	4.43	100	17.64	8.89	0.18	6.11	0.35

C Appendix SVBRDF Recovery

C.1 Derivation of Convolution Model

The outgoing radiance at point p along direction ω_o, $L_o(p, \omega_o)$ is given by

$$B(p, \omega_o) = \int_{H^2(\mathbf{n})} f_r(p, \omega_o, \omega_i) L(p, \omega_i) \cos\theta_i d\omega_i, \tag{C.1}$$

where f is the BRDF and $L(p, \omega_i)$ is the incident radiance along direction ω_i. For the Torrance-Sparrow BRDF, f is defined as

$$f(p, \omega_o, \omega_i) = K_d + K_s \frac{D(\omega_m) F(\omega_o \cdot \omega_m) G(\omega_i, \omega_o)}{4 \cos\theta_i \cos\theta_o}, \tag{C.2}$$

where ω_m is the half-direction vector $\omega_m = (\omega_i + \omega_o)/||\omega_i + \omega_o||$; $D(\omega_m)$ is the normal distribution function; $F(\omega_o \cdot \omega_m)$ is the Fresnel term. Ramamoorthi and Hanrahan simplify this term to $F(\theta_o)$, as the angle θ_o is often close to the angle between ω_o and ω_m; $G(\omega_i, \omega_o)$ is the shadowing-masking term. Ramamoorthi and Hanrahan ignore G. We assume shadowing and masking are independent statistical events, so that $G(\omega_i, \omega_o) = G(\omega_i)G(\omega_o)$.

There are two important notes about the denominator in Equation C.2:

1. $1/(4\cos\theta_o)$ results from the half-direction transform: the distribution of microfacets with a normal ω_m is transformed to the distribution of outgoing directions ω_o that the incoming light ray ω_i reflects toward (see Pharr et al. [182], Equation 9.27).
2. $1/(\cos\theta_i)$ cancels out the cosine term applied to the incoming radiance (see Pharr et al. [182], equation 9.30).

We now substitute Equation C.2 into Equation C.1 and split the equation into diffuse and specular

$$\begin{aligned} B(p, \omega_o) = K_d \int_{H^2(\mathbf{n})} L(p, \omega_i) \cos\theta_i d\omega_i \\ + K_s \int_{H^2(\mathbf{n})} \frac{D(\omega_m) F(\omega_o \cdot \omega_m) G(\omega_i, \omega_o)}{4 \cos\theta_i \cos\theta_o} L(p, \omega_i) \cos\theta_i d\omega_i \end{aligned} \tag{C.3}$$

This equation is simplified using the assumptions that F only depends on θ_o and shadowing-masking is ignored. We replace the

integral of incoming radiance with the symbol for irradiance E.

$$B(p, \omega_o) = K_d E(p) + K_s F(\theta_o) \int_{H^2(\mathbf{n})} \frac{D(\omega_m)}{4 \cos \theta_o} L(p, \omega_i) d\omega_i. \quad \text{(C.4)}$$

Ramamoorthi and Hanrahan rewrite the specular term as a convolution between a filter based on D, and L. Crucially, the domain of D in the Torrance-Sparrow model is the half-angle space. In Ramamoorthi and Hanrahan's derivation, the spherical harmonic representation for this filter, in the paper referred to as S is derived in incoming-direction space for normal exitance (Ramamoorthi and Hanrahan, Equation 27). This has two consequences:

1. We do not have to account for a change of variables and $1/(4 \cos \theta_o)$ can be removed.
2. In reality, S depends on the outgoing direction that is observed and thus, the filter changes shape. This variation is ignored with the explanation that "the BRDF filter is essentially symmetric about the reflected direction for small viewing angles, as well as for low frequencies l. Hence, it can be shown by Taylor-series expansions and verified numerically, that the corrections to equation 20 [Equation 5.9 in our paper] are small under these conditions."

This means that we can rewrite Equation C.4 with a convolution

$$B(p, \omega_o) = K_d E(p) + K_s F(\theta_o) \left[S * L \right]_{\omega_o}, \quad \text{(C.5)}$$

which equals Equations 21 and 22 in Ramamoorthi and Hanrahan.

C.2 Sampling Theory

The transformation from the directional domain to spherical harmonics begs the question: do we have the enough samples to accurately recover the coefficients of the outgoing radiance? We know from Equation 5.9 that the BRDF acts as a low-pass filter parameterized by α. We connect this knowledge with sampling theory to derive lower bounds on sampling counts.

The Nyquist-Shannon theorem provides a lower bound on the number of samples required to exactly recover a band-limited signal using a Fourier series. Similar theorems have been developed for spherical harmonics [157, 158, 51]. These state that, to recover a spherical signal with band-limit ℓ^*, the number of samples should be $\mathcal{O}(\ell^{*2})$. The sampling rate and related band-limit have direct consequences for BRDF recovery. Assume that the incoming light has been sampled at a high enough rate to be accurately recovered, for example, from projected photographs or a gazing sphere. Then the outgoing light is the weakest link, as it is sampled by moving the camera along N positions around the object. Sampling theory

tells us that we can only accurately recover outgoing radiance that is band-limited to $\ell^* < \sqrt{N}$ degrees. Signals with non-zero amplitude in higher degrees will suffer from aliasing.

Fortunately, the BRDF acts as a low-pass filter on the incoming radiance (Equation 5.9). That means the outgoing radiance can fall into two categories, based on the α parameter of the material (α = roughness^2): α is either too low or α is high enough to recover spherical harmonic coefficients. If α is too low, the low-pass filtering from the BRDF does not band-limit the signal enough to accurately recover with the given sampling rate. The threshold for α can be determined based on Equation 5.9. Let t be an acceptable attenuation factor for degrees $\ell > \ell^*$. We solve Equation 5.9 for t to find the lower bound, α', for accurate recovery

$$\alpha' = \ell^{*-1}\sqrt{-\ln t}. \tag{C.6}$$

An acceptable threshold t can be determined empirically, by investigating the reconstruction error for a set of environment maps. To provide some intuition, for $N = 400$ samples and a threshold of $t = 0.5$, $\alpha' \approx 0.07$. Above this threshold, our method can recover α and K_s to an acceptable accuracy, provided that the incoming radiance has enough amplitude in the right degrees. This also extends to non-uniform samples, because the Nyquist-Shannon theorem holds for non-uniform samples [154]. In other words: if a lower bound on α is known, it does not matter where the camera is placed, as long as the average distance to the closest sample is equal to $1/N$. It also means that one can determine the number of required views based on the lowest α that should be recovered: $N \sim \alpha^{-2}$.

It is important to understand what happens if $\alpha < \alpha'$. First, we are uncertain when α lands between 0 and α', based on the power spectrum alone. For $0 < \alpha < \alpha'$, Equation 5.9 is close to 1 for all degrees below ℓ^*. Second, because this situation occurs for low α, the outgoing radiance should be similar to the incoming radiance, up to a scaling factor for absorption and transmission. It is unlikely that the spherical harmonics decomposition with significant aliasing will match that of the incoming radiance. Therefore, we can detect that $\alpha < \alpha'$: then, the MSE for any parameter combination ψ is high. Once such a case is detected, we know that our spherical harmonic-based analysis provides no further insights on (un)certainty. There is still a chance for accurate BRDF recovery if $\alpha < \alpha'$. This is the case when there is high local variation in the incoming light around the sample locations, resulting in large changes in radiance for small changes in α. One could quantify this variation by comparing the difference between the sample location for $\alpha = 0$ and $\alpha = \alpha'$ and use this as a measure of certainty. In our work, we find that our certainty measure works well, even for $\alpha < \alpha'$ and thus, we do not add this measure.

C.3 Stanford ORB other results

We include the results from Stanford ORB in Table C.1 for reference. These results were obtained under different acquisition condition and cannot be directly compared to our results.

	PSNR-H↑	PSNR-L↑	SSIM↑	LPIPS↓
NVDiffRecMC [86] †	25.08	32.28	0.974	0.027
NVDiffRec [164] †	24.93	32.42	0.975	0.027
PhySG [266]	21.81	28.11	0.960	0.055
NVDiffRec [164]	22.91	29.72	0.963	0.039
NeRD [14]	23.29	29.65	0.957	0.059
NeRFactor [267]	23.54	30.38	0.969	0.048
InvRender [248]	23.76	30.83	0.970	0.046
NVDiffRecMC [86]	24.43	31.60	0.972	0.036
Neural-PBIR [219]	26.01	33.26	**0.979**	**0.023**

Table C.1: Benchmark Comparison for **Novel Scene Relighting** of Existing Methods from [120]. † denotes models trained with the ground-truth 3D scans and pseudo materials optimized from light-box captures. The rest of results are obtained by optimizing jointly for illumination, geometry and material. *We report these numbers for reference, however they cannot be directly compared to our results.*

C.4 Ablations

C.4.1 Spherical Harmonics fitting

Our method computes spherical harmonic coefficients using a least-squares fit and includes a regularizer. We would like to understand the effect of the maximum degree that is estimated, find the optimal weight for the regularizer, and see if the regularizer has the desired effect (improved accuracy). The results for the maximum degree are presented in Table C.2. We find that more degrees help, but also that we obtain good results with a relatively low number of degrees (from 3 on). In Table C.3, we find that our approach is not very sensitive to the specific setting of the regularizer, with an optimal value near $\lambda = 1 \times 10^{-4}$. When optimizing BRDF parameters, it is typical to weight samples based on their angle with the normal, θ. For example, samples at grazing angles are often associated with lower confidence and weighted less than samples near $\theta = 0$. We set up a general weighting function, $\max(0, 1 - (1 - \cos a\theta)^b)$ that ignores samples with $\theta > a\frac{\pi}{2}$ and weights the rest with a smooth falloff determined by b. We observe in Table C.4 that $a = 1, b = 1$ gives the best results. We also observe that weighting is beneficial, compared to constant weight (top row).

Table C.2: Ablation max degree ℓ^*

ℓ^*	PSNR-H↑	PSNR-L↑	SSIM↑	LPIPS↓	Time
0	25.670	32.580	0.971	0.042	0.94s
1	25.531	32.536	0.971	0.042	1.08s
2	25.588	32.568	0.971	0.043	1.10s
3	25.881	32.983	0.972	0.041	1.13s
4	26.134	33.142	0.972	0.040	1.21s
5 (Ours)	26.182	33.215	0.972	0.040	1.42s
6	26.199	33.266	0.972	0.040	1.70s
7	26.147	33.227	0.972	0.040	2.05s
8	26.153	33.241	0.972	0.040	3.02s

Table C.3: Ablation regularizer weight

λ	PSNR-H↑	PSNR-L↑	SSIM↑	LPIPS↓	Time
1×10^{-2}	26.498	33.683	0.976	0.033	**4.32**s
1×10^{-3}	26.524	33.703	0.976	0.033	4.33s
1×10^{-4}	26.582	33.762	0.976	0.033	4.33s
1×10^{-5}	26.484	33.667	0.975	0.033	4.33s
1×10^{-6}	**26.638**	**33.807**	**0.976**	**0.033**	4.33s
1×10^{-6} constant	26.611	33.788	0.976	**0.032**	4.36s
1×10^{-7}	26.585	33.750	0.975	0.033	4.33s

Table C.4: Ablation sample weighting

$\max(0, 1 - (1 - \cos a\theta)^b)$	PSNR-H↑	PSNR-L↑	SSIM↑	LPIPS↓	Time
No weighting	26.457	33.607	0.975	0.034	4.33s
$a = 0.8, b = 1$	26.445	33.705	0.976	**0.032**	4.33s
$a = 0.9, b = 1$	26.529	33.725	0.976	0.032	4.33s
$a = 1, b = 1$ (Ours)	**26.638**	**33.807**	**0.976**	**0.033**	4.33s
$a = 1, b = 2$	26.618	33.800	0.976	0.032	4.33s
$a = 1, b = 3$	26.588	33.778	0.976	0.033	4.33s
$a = 1, b = 4$	26.558	33.750	0.975	0.033	4.33s
$a = 1, b = 5$	26.530	33.719	0.975	0.033	4.33s

Curriculum Vitæ | D

Ruben Wiersma

7 September 1994
Born in Leiden, the Netherlands.

Education

2005–2011
Visser 't Hooft Lyceum Leiden, the Netherlands

VWO Gymnasium, *cum laude*

2013–2024
TU Delft, the Netherlands

Propedeuse Industrial Design Engineering, *cum laude* ('13–'14)
BSc, Computer Science, *cum laude* ('14–'17)
MSc, Computer Science, *cum laude* ('17–'19)
PhD, Computer Graphics ('19-'24)

E Publications

7. **R. Wiersma**, A. Nasikun, E. Eisemann, and K. Hildebrandt. (2023). *A Fast Geometric Multigrid Method for Curved Surfaces.* ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23).

6. L. Tissen, S. Frequin, **R. Wiersma**. (2023). *The case of the golden background, a virtual restoration and a physical reconstruction of the medieval Crucifixion of the Lindau Master (c. 1425).* Digital Humanities Quarterly, 17, 1.

5. J. van der Toorn, **R. Wiersma**, A. Vandivere, R. Marroquim, E. Eisemann. (2022). *A New Baseline for Feature Description on Multimodal Imaging of Paintings.* 2022 Eurographics Workshop on Graphics and Cultural Heritage.

4. **R. Wiersma**, A. Nasikun, E. Eisemann, K. Hildebrandt. (2022). *DeltaConv: anisotropic operators for geometric deep learning on point clouds.* ACM Transactions on Graphics (ToG) 41, 4, Article 105 (SIGGRAPH 2022).

3. Y. Lin, **R. Wiersma**, S. L. Pintea, K. Hildebrandt, E. Eisemann, J. C. van Gemert. (2022). *Deep vanishing point detection: Geometric priors make dataset variations vanish.* 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

2. J. Wempe., R. van den Brink, E. Mooldijk, N. Feirabend, **R. Wiersma**, J. Sietsma, J. Dik. (2020). *Revealing unique inscriptions of a Nazi collaborator in Doodencel 601 of the Oranjehotel.* Heritage Science 8, 74.

1. **R. Wiersma**, E. Eisemann and K. Hildebrandt. (2020). *CNNs on Surfaces using Rotation-Equivariant Features.* ACM Transactions on Graphics (ToG) 39, 4, Article 92 (SIGGRAPH 2020).

Bibliography

[1] Adobe. 2016. Adobe Mixamo 3D characters. www.mixamo.com

[2] Miika Aittala, Timo Aila, and Jaakko Lehtinen. 2016. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (ToG)* 35, 4, Article 65 (2016).

[3] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. 2013. Practical SVBRDF capture in the frequency domain. *ACM Transactions on Graphics (ToG)* 32, 4 (2013), 110–1.

[4] Miika Aittala, Tim Weyrich, Jaakko Lehtinen, et al. 2015. Two-shot SVBRDF capture for stationary materials. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 110–1.

[5] Burak Aksoylu, Andrei Khodakovsky, and Peter Schröder. 2005. Multilevel Solvers for Unstructured Surface Meshes. *SIAM Journal on Scientific Computing* 26, 4 (2005), 1146–1165.

[6] Marc Alexa and Max Wardetzky. 2011. Discrete Laplacians on General Polygonal Meshes. *ACM Transactions on Graphics (ToG)* 30, 4 (2011), 102:1–102:10.

[7] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: Shape Completion and Animation of People. *ACM Transactions on Graphics (ToG)* 24, 3 (2005), 408–416.

[8] Matan Atzmon, Haggai Maron, Yaron Lipman, Atzmon Matan, Maron Haggai, and Lipman Yaron. 2018. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (ToG)* 37, 4 (2018), 71:1–71:12.

[9] Nathan Bell, Luke N. Olson, and Jacob Schroder. 2022. PyAMG: Algebraic Multigrid Solvers in Python. *Journal of Open Source Software* 7, 72 (2022), 4142.

[10] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 2018. 3DmFV: Three-Dimensional Point Cloud Classification in Real-Time Using Convolutional Neural Networks. *IEEE Robotics and Automation Letters* 3 (2018), 3145–3152.

[11] Sai Bi, Zexiang Xu, Pratul P. Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Milos Hasan, Yannick Hold-Geoffroy, David J. Kriegman, and Ravi Ramamoorthi. 2020. Neural Reflectance Fields for Appearance Acquisition. *arXiv preprint arXiv:2008.03824* (2020).

[12] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014. FAUST: Dataset and evaluation for 3D mesh registration. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).

[13] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. 2016. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. 3189–3197.

[14] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. 2021. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12684–12694.

[15] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan Barron, and Hendrik Lensch. 2021. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. *Advances in Neural Information Processing Systems* 34 (2021), 10691–10704.

[16] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik Lensch, and Jan Kautz. 2020. Two-shot spatially-varying brdf and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3982–3991.

[17] Alexandre Boulch. 2020. ConvPoint: Continuous convolutions for point cloud processing. *Computer Graphics Forum* 88 (2020), 24–34.

[18] James H. Bramble. 1993. *Multigrid Methods*. Chapman and Hall/CRC.

[19] Achi Brandt. 1986. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.* 19, 1 (1986), 23–56.

[20] Christopher Brandt and Klaus Hildebrandt. 2017. Compressed Vibration Modes of Deformable Bodies. *Computer Aided Geometric Design* 52–53 (2017), 297–312.

[21] Alexander M. Bronstein, Michael M. Bronstein, Leonidas J. Guibas, and Maks Ovsjanikov. 2011. Shape Google: Geometric Words and Expressions for Invariant Shape Retrieval. 30, 1 (2011), 1:1–1:20.

[22] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. 2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv preprint arXiv:2104.13478* (2021).

[23] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[24] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*.

[25] Brent Burley and Walt Disney Animation Studios. 2012. Physically-based shading at disney. In *ACM SIGGRAPH*, Vol. 2012. vol. 2012, 1–7.

[26] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012* (2015).

[27] Wesley Chang, Venkataram Sivaram, Derek Nowrouzezahrai, Toshiya Hachisuka, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Parameter-space ReSTIR for Differentiable and Inverse Rendering. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA.

[28] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z Chen, and Jian Wu. 2020. A Hierarchical Graph Network for 3D Object Detection on Point Clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[29] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. , 22:1–22:14 pages.

[30] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[31] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. 2018. Spherical CNNs. In *International Conference on Learning Representations (ICLR)*.

[32] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. 2019. Gauge Equivariant Convolutional Networks and the Icosahedral CNN. *International Conference on Machine Learning (ICML)* (2019).

[33] Taco Cohen and Max Welling. 2016. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*. 2990–2999.

[34] Taco Cohen and Max Welling. 2017. Steerable CNNs. In *International Conference on Learning Representations (ICLR)*.

[35] Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schroder. 2013. Digital Geometry Processing with Discrete Exterior Calculus. *SIGGRAPH Asia 2013 Courses* (2013), 7:1–7:126.

[36] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (ToG)* 32, 5 (2013), 152.

[37] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner.

2017. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[38] Timothy A. Davis, Sivasankaran Rajamanickam, and Wissam M. Sid-Lakhdar. 2016. A survey of direct methods for sparse linear systems. *Acta Numerica* 25 (2016), 383–566.

[39] Fernando de Goes, Mathieu Desbrun, and Yiying Tong. 2016. Vector Field Processing on Triangle Meshes. *SIGGRAPH Asia 2016 Courses* (2016), 27:1–27:49.

[40] Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. 2021. Gauge Equivariant Mesh CNNs: Anisotropic convolutions on geometric graphs. *International Conference on Learning Representations (ICLR)* (2021).

[41] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*. 3844–3852.

[42] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. 2021. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[43] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)* 37, 4 (2018), 1–15.

[44] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. 2019. Flexible svbrdf capture with a multi-image deep network. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 1–13.

[45] Valentin Deschaintre, George Drettakis, and Adrien Bousseau. 2020. Guided Fine-Tuning for Large-Scale Material Transfer. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 39, 4 (2020).

[46] Valentin Deschaintre, Yiming Lin, and Abhijeet Ghosh. 2021. Deep polarization imaging for 3D shape and SVBRDF acquisition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15567–15576.

[47] Christian Dick, Marcus Rogowsky, and Rüdiger Westermann. 2016. Solving the Fluid Pressure Poisson Equation Using Multigrid - Evaluation and Improvements. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 22, 11 (2016), 2480–2492.

[48] Miguel Dominguez, Rohan Dhamdhere, Atir Petkar, Saloni Jain, Shagan Sah, and Raymond Ptucha. 2018. General-Purpose Deep Point Cloud Feature Extractor. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2018).

[49] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. 2014. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Transactions on Graphics (ToG)* 33, 6 (2014), 1–12.

[50] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*.

[51] James R. Driscoll and Dennis M. Healy. 1994. Computing Fourier Transforms and Convolutions on the 2-Sphere. *Advances in Applied Mathematics* 15, 2 (1994), 202–250.

[52] Jonathan Dupuy and Wenzel Jakob. 2018. An Adaptive Parameterization for Efficient Material Acquisition and Rendering. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 274:1–274:18.

[53] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. 2005. A frequency analysis of light transport. *ACM Transactions on Graphics (ToG)* 24, 3 (2005), 1115–1126.

[54] Moshe Eliasof and Eran Treister. 2020. DiffGCN: Graph Convolutional Networks via Differential Operators and Algebraic Multigrid Pooling. *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[55] Andreas Engelhardt, Amit Raj, Mark Boss, Yunzhi Zhang, Abhishek Kar, Yuanzhen Li, Deqing Sun, Ricardo Martin Brualla, Jonathan T. Barron, Hendrik P. A. Lensch, and Varun Jampani. 2024. SHINOBI: SHape and Illumination using Neural Object Decomposition via BRDF Optimization In-the-wild. *ArXiv e-prints* (2024).

[56] Martin Erwig. 2000. The graph Voronoi diagram with applications. *Networks* 36, 3 (2000), 156–163.

[57] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 2017. Learning SO(3) Equivariant Representations with Spherical CNNs. *European Conference on Computer Vision (ECCV)* (2017).

[58] Danielle Ezuz, Justin Solomon, Vladimir G. Kim, and Mirela Ben-Chen. 2017. GWCNN: A Metric Alignment Layer for Deep Shape Analysis. *Computer Graphics Forum* 36, 5 (2017), 49–57.

[59] Bianca Falcidieno. 2007. Bringing the Semantics into Digital Shapes: the AIM@SHAPE Approach. In *Eurographics Italian Chapter Conference*, Raffaele De Amicis and Giuseppe Conti (Eds.). The Eurographics Association.

[60] Chongrui Fan, Yiming Lin, and Abhijeet Ghosh. 2023. Deep Shape and SVBRDF Estimation using Smartphone Multi-lens Imaging. *Computer Graphics Forum* (2023).

[61] Beat Fasel and Daniel Gatica-Perez. 2006. Rotation-invariant neoperceptron. In *International Conference on Pattern Recognition (ICPR)*, Vol. 3. IEEE, 336–339.

[62] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. *AAAI Conference on Artificial Intelligence (AAAI)* (2019).

[63] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[64] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. 2018. SplineCNN: Fast Geometric Deep Learning With Continuous B-Spline Kernels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 869–877.

[65] Roland W. Fleming, Ron O. Dror, and Edward H. Adelson. 2003. Real-world illumination and the perception of surface reflectance properties. *Journal of Vision* 3, 5 (2003), 3–3.

[66] Chamberlain Fong. 2015. Analytical Methods for Squaring the Disc. *arXiv preprint arXiv:1509.06344* (2015).

[67] William T. Freeman and Edward H Adelson. 1991. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (1991), 891–906.

[68] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. 2020. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[69] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*. PMLR, 1050–1059.

[70] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. 2019. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Transactions on Graphics (ToG)* 38, 4 (2019), 134–1.

[71] Joachim Georgii and Rüdiger Westermann. 2006. A multigrid framework for real-time simulation of deformable bodies. *Computers & Graphics* 30, 3 (2006), 408–415.

[72] Jan E. Gerken, Jimmy Aronsson, Oscar Carlsson, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. 2021. Geometric deep learning and equivariant neural networks. *arXiv preprint arXiv:2105.13926* (2021).

[73] Abhijeet Ghosh, Shruthi Achutha, Wolfgang Heidrich, and Matthew O'Toole. 2007. BRDF Acquisition with Basis Illumination. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, Rio de Janeiro, Brazil, 1–8.

[74] Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. 2024. Bayes' Rays: Uncertainty Quantification in Neural Radiance Fields. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024).

[75] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. 2021. Revisiting Point Cloud Shape Classification with a Simple and Effective Baseline. *International Conference on Machine Learning (ICML)* (2021).

[76] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 2018. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[77] Seth Green, George Turkiyyah, and Duane W. Storti. 2002. Subdivision-based multilevel methods for large scale engineering simulation of thin shells. In *ACM Symposium on Solid Modeling and Applications*. ACM, 265–272.

[78] Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. https://eigen.tuxfamily.org

[79] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. 2018. PCPNet Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum* 37, 2 (2018), 75–85.

[80] Jie Guo, Shuichang Lai, Chengzhi Tao, Yuelong Cai, Lei Wang, Yanwen Guo, and Ling-Qi Yan. 2021. Highlight-aware two-stream network for single-image SVBRDF acquisition. *ACM Transactions on Graphics (ToG)* 40, 4, Article 123 (2021).

[81] Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. 2020. Materialgan: reflectance capture using a generative svbrdf model. *arXiv preprint arXiv:2010.00114* (2020).

[82] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2020. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43 (2020), 4338–4364.

[83] Wolfgang Hackbusch. 1985. *Multi-grid methods and applications*. Springer series in computational mathematics, Vol. 4. Springer.

[84] Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. 2019. Surface Networks via General Covers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 632–641.

[85] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics (ToG)* 38, 4 (2019), 90.

[86] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, light, and material decomposition from images using Monte Carlo rendering and denoising. *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), 22856–22869.

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[88] Philipp Henzler, Valentin Deschaintre, Niloy J Mitra, and Tobias Ritschel. 2021. Generative Modelling of BRDF Textures from Flash Images. *ACM Transactions on Graphics (ToG)* 40, 6 (2021).

[89] Philipp Herholz and Marc Alexa. 2018. Factor once: reusing cholesky factorizations on sub-meshes. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 230.

[90] Philipp Herholz and Marc Alexa. 2019. Efficient Computation of Smoothed Exponential Maps. *Computer Graphics Forum* 38, 6 (2019), 79–90.

[91] Philipp Herholz and Olga Sorkine-Hornung. 2020. Sparse cholesky updates for interactive mesh parameterization. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 202:1–202:14.

[92] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. 2018. Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 235:1–235:12.

[93] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*. Springer, 44–51.

[94] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), 6840–6851.

[95] Hugues Hoppe. 1996. Progressive Meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, John Fujii (Ed.). ACM, 99–108.

[96] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. 2016. SceneNN: A Scene Meshes Dataset with aNNotations. *International Conference on 3D Vision (3DV)* (2016).

[97] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. 2018. Pointwise Convolutional Neural Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[98] Jingwei Huang, Haotian Zhang, Li Yi, Thomas A. Funkhouser, Matthias Nießner, and Leonidas J. Guibas. 2019. TextureNet: Consistent Local Parametrizations for Learning From High-Resolution Signals on Meshes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4440–4449.

[99] Inseung Hwang, Daniel S. Jeon, Adolfo Muñoz, Diego Gutierrez, Xin Tong, and Min H. Kim. 2022. Sparse Ellipsometry: Portable Acquisition of Polarimetric SVBRDF and Shape with Unstructured Flash Photography. *ACM Transactions on Graphics (ToG)* 41, 4 (2022).

[100] Intel Corporation. 2023. oneMKL PARDISO - Parallel Direct Sparse Solver Interface. https://www.intel.com/content/www/us/en/docs/onemkl/developer-reference-c/2023-1/onemkl-pardiso-parallel-direct-sparse-solver-iface.html. Accessed: April 17, 2023.

[101] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning (ICML)* (2015).

[102] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022. *Mitsuba 3 renderer*. https://mitsuba-renderer.org

[103] Edwin T Jaynes. 1957. Information theory and statistical mechanics. *Physical review* 106, 4 (1957), 620.

[104] In-Yong Jeon, Kwang-Jin Choi, Tae-Yong Kim, Bong-Ouk Choi, and Hyeong-Seok Ko. 2013. Constrainable Multigrid for Cloth. *Computer Graphics Forum* 32, 7 (2013), 31–39.

[105] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Nießner. 2019. Spherical CNNs on Unstructured Grids. *International Conference on Learning Representations (ICLR)* (2019).

[106] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. 2023. TensoIR: Tensorial Inverse Rendering. *arXiv preprint arXiv:2304.12461* (2023).

[107] Misha Kazhdan and Hugues Hoppe. 2019. An Adaptive Multi-Grid Solver for Applications in Computer Graphics. *Computer Graphics Forum* 38, 1 (2019), 138–150.

[108] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. 2012. Can mean-curvature flow be modified to be non-singular?. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1745–1754.

[109] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Symposium on Geometry Processing (ACM International Conference Proceeding Series)*, Vol. 256. Eurographics Association, 61–70.

[110] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics (ToG)* 42, 4 (2023).

[111] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

[112] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[113] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics (ToG)* 36, 4 (2017).

[114] Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. 1998. A General Framework for Mesh Decimation. In *Proceedings of the Graphics Interface 1998 Conference, June 18-20, 1998, Vancouver, BC, Canada*, Wayne A. Davis, Kellogg S. Booth, and Alain Fournier (Eds.). Canadian Human-Computer Communications Society, 43–50.

[115] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. 2018. Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network. In *Advances in Neural Information Processing Systems (NeurIPS)*. 10138–10147.

[116] Ralf Kornhuber and Harry Yserentant. 2008. Multigrid methods for discrete elliptic problems on triangular surfaces. *Computing and Visualization in Science* 11, 4 (2008), 251–257.

[117] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. 2018. Surface Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[118] Dilip Krishnan, Raanan Fattal, and Richard Szeliski. 2013. Efficient preconditioning of laplacian matrices for computer graphics. *ACM Transactions on Graphics (ToG)* 32, 4 (2013), 142:1–142:15.

[119] Dilip Krishnan and Richard Szeliski. 2011. Multigrid and multilevel preconditioners for computational photography. *ACM Transactions on Graphics (ToG)* 30, 6 (2011), 177.

[120] Zhengfei Kuang, Yunzhi Zhang, Hong-Xing Yu, Samir Agarwala, Shangzhe Wu, and Jiajun Wu. 2023. Stanford-ORB: A Real-World 3D Object Inverse Rendering Benchmark. *arXiv preprint arXiv:2310.16044* (2023).

[121] Alon Lahav and A. Tal. 2020. MeshWalker: deep mesh understanding by random walks. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 263:1–263:13.

[122] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics* 39, 6 (2020).

[123] Shiyi Lan, Ruichi Yu, Gang Yu, and L Davis. 2019. Modeling Local Geometric Structure of 3D Point Clouds Using Geo-CNN. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[124] Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. 2016. TI-POOLING: Transformation-invariant pooling for feature learning in convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 289–297.

[125] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. 2007. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning (ICML)*. ACM, 473–480.

[126] Eric-Tuan Le, Iasonas Kokkinos, and Niloy J Mitra. 2020. Going Deeper With Lean Point Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[127] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.

[128] Huan Lei, Naveed Akhtar, and Ajmal Mian. 2019. Octree guided CNN with Spherical Kernels for 3D Point Clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[129] Hendrik Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. 2003. Image-based Reconstruction of Spatial Appearance and Geometric Detail. *ACM Transactions on Graphics (ToG)* 22 (2003), 234–257.

[130] Hendrik P.A. Lensch, Jochen Lang, Asla M. Sá, and Hans-Peter Seidel. 2003. Planned Sampling of Spatially Varying BRDFs. *Computer Graphics Forum* 22, 3 (2003), 473–482.

[131] Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2017. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–11.

[132] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution On X-Transformed Points. In *Advances in Neural Information Processing Systems (NeurIPS)*. 820–830.

[133] Yangyan Li, Bu Rui, Mingchao Sun, Wei Wu, Xinhan Di, Baoquan Chen, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution on x-transformed points. *Advances in Neural Information Processing Systems (NeurIPS)* (2018).

[134] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2018. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 1–11.

[135] Zhouhui Lian, Afzal Godil, Benjamin Bustos, Mohamed Daoudi, Jeroen Hermans, Shun Kawamura, Yukinori Kurita, Guillaume Lavoué, Hien Nguyen, Ryutarou Ohbuchi, Yuki Ohkita, Yuya Ohishi, Fatih Porikli, Martin Reuter, Ivan Sipiran, Dirk Smeets, Paul Suetens, Hedi Tabia, and Dirk Vandermeulen. 2011. SHREC '11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes. *Eurographics Workshop on 3D Object Retrieval*, 79–88.

[136] Jian Liang and Hongkai Zhao. 2013. Solving Partial Differential Equations on Point Clouds. *SIAM Journal on Scientific Computing* 35 (2013), A1461–A1486.

[137] Liqiang Lin, Pengdi Huang, Chi-Wing Fu, Kai Xu, Hao Zhang, and Hui Huang. 2020. One Point is All You Need: Directional Attention Point for Feature Learning. *arXiv preprint arXiv:2012.06257* (2020).

[138] Or Litany, Alexander M. Bronstein, Michael M. Bronstein, and Ameesh Makadia. 2018. Deformable Shape Completion with Graph Convolutional Autoencoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1886–1895.

[139] Hsueh-Ti Derek Liu, Alec Jacobson, and Keenan Crane. 2017. A Dirac Operator for Extrinsic Shape Analysis. *Computer Graphics Forum* 36, 5 (2017), 139–149.

[140] Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface Multigrid via Intrinsic Prolongation. *ACM Transactions on Graphics (ToG)* 40, 4 (2021).

[141] Jinxian Liu, Bingbing Ni, Caiyuan Li, Jiancheng Yang, and Qi Tian. 2019. Dynamic Points Agglomeration for Hierarchical Point Sets Learning. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2019).

[142] Kun Liu, Qing Wang, Wolfgang Driever, and Olaf Ronneberger. 2012. 2d/3d rotation-invariant detection using equivariant filters and kernel weighted mapping. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 917–924.

[143] Weiping Liu, Jia Sun, Wanyi Li, Ting Hu, and Peng Wang. 2019. Deep Learning on Point Clouds and Its Application: A Survey. *Sens* 19 (2019), 4188.

[144] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. 2019. DensePoint: Learning densely contextual representation for efficient point cloud processing. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2019).

[145] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. 2019. Relation-Shape Convolutional Neural Network for Point Cloud Analysis. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[146] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. 2020. A Closer Look at Local Aggregation Operators in Point Cloud Analysis. *European Conference on Computer Vision (ECCV)* (2020).

[147] Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. *International Conference on Learning Representations (ICLR)* (2017).

[148] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Transactions on Graphics (ToG)* 38, 6 (2019).

[149] Tao Lu, Limin Wang, and Gangshan Wu. 2021. CGA-Net: Category Guided Aggregation for Point Cloud Semantic Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

[150] Shi Mao, Chenming Wu, Zhelun Shen, and Liangjun Zhang. 2023. NeuS-PIR: Learning Relightable Neural Surface using Pre-Integrated Rendering. *arXiv preprint arXiv:2306.07632* (2023).

[151] Diego Marcos, Michele Volpi, and Devis Tuia. 2016. Learning rotation invariant convolutional filters for texture classification. In *International Conference on Pattern Recognition (ICPR)*. IEEE, 2012–2017.

[152] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 71.

[153] Rosalie Martin, Arthur Roullier, Romain Rouffet, Adrien Kaiser, and Tamy Boubekeur. 2022. MaterIA: Single Image High-Resolution Material Capture in the Wild. *Computer Graphics Forum (Proc. EUROGRAPHICS 2022)* to appear, to appear (2022), to appear.

[154] Farokh Marvasti. 2012. *Nonuniform sampling: theory and practice*. Springer Science & Business Media.

[155] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 37–45.

[156] Aleka McAdams, Eftychios Sifakis, and Joseph Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids. In *Symposium on Computer Animation (SCA)*. Eurographics Association, 65–73.

[157] Jason D. McEwen, Gilles Puy, Jean-Philippe Thiran, Pierre Vandergheynst, Dimitri Van De Ville, and Yves Wiaux. 2011. Sampling theorems and compressive sensing on the sphere. In *Wavelets and Sparsity XIV*, Manos Papadakis, Dimitri Van De Ville, and Vivek K. Goyal (Eds.), Vol. 8138. International Society for Optics and Photonics, SPIE, 81381F.

[158] Jason D. McEwen and Yves Wiaux. 2011. A novel sampling theorem on the sphere. *IEEE Transactions on Signal Processing* 59, 12 (2011), 5876–5887.

[159] Eivind Lyche Melvær and Martin Reimers. 2012. Geodesic polar coordinates on polygonal meshes. In *Computer Graphics Forum (CGF)*, Vol. 31. 2423–2435.

[160] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. 2020. Primal-Dual Mesh Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[161] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision (ECCV)*.

[162] Thomas W. Mitchel, Vladimir G. Kim, and Michael Kazhdan. 2021. Field Convolutions for Surface CNNs. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2021).

[163] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1. IEEE, 3.

[164] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas

Müller, and Sanja Fidler. 2022. Extracting triangular 3d models, materials, and lighting from images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8280–8290.

[165] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H. Kim. 2018. Practical SVBRDF Acquisition of 3D Objects with Unstructured Flash Photography. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 267:1–12.

[166] Ahmad Nasikun and Klaus Hildebrandt. 2022. The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems. *ACM Transactions on Graphics (ToG)* 41, 2 (2022), 17:1–17:14.

[167] Andrew Nealen. 2004. An As-Short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. *URL: http://www.nealen.com/projects/mls/asapmls.pdf* 1 (2004).

[168] Xinlai Ni, Michael Garland, and John C. Hart. 2004. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics (ToG)* 23, 3 (2004), 613–622.

[169] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Transactions on Graphics (ToG)* 40, 6 (2021).

[170] Baptiste Nicolet, Fabrice Rousselle, Jan Novák, Alexander Keller, Wenzel Jakob, and Thomas Müller. 2023. Recursive Control Variates for Inverse Rendering. *ACM Transactions on Graphics (ToG)* 42, 4 (2023).

[171] Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. 2021. Material and Lighting Reconstruction for Complex Indoor Scenes with Texture-space Differentiable Rendering. In *Eurographics Symposium on Rendering - DL-only Track*, Adrien Bousseau and Morgan McGuire (Eds.). The Eurographics Association.

[172] Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. 2022. Unbiased Inverse Volume Rendering with Differential Trackers. *ACM Transactions on Graphics (ToG)* 41, 4, Article 44 (2022), 44:1–44:20 pages.

[173] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Transactions on Graphics (ToG)* 39, 4 (2020).

[174] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Transactions on Graphics (ToG)* 38, 6 (2019).

[175] Barrett O'Neill. 1983. *Semi-Riemannian Geometry With Applications to Relativity*. Elsevier Science.

[176] Miguel A. Otaduy, Daniel Germann, Stephane Redon, and Markus H. Gross. 2007. Adaptive deformations with fast tight bounds. In *Symposium on Computer Animation (SCA)*, Michael Gleicher and Daniel Thalmann (Eds.). Eurographics Association, 181–190.

[177] Guanghua Pan, Jun Wang, Rendong Ying, and Peilin Liu. 2018. 3DTI-Net: Learn Inner Transform Invariant 3D Geometry Features using Dynamic GCN. *arXiv preprint arXiv:1812.06254* (2018).

[178] Hao Pan, Shilin Liu, Yang Liu, and Xin Tong. 2018. Convolutional Neural Networks on 3D Surfaces Using Parallel Frames. *arXiv preprint arXiv:1808.04952* (2018).

[179] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[180] Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Transactions on Graphics (ToG)* 22, 3 (2003), 313–318.

[181] Pietro Perona and Jitendra Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 12, 7 (1990), 629–639.

[182] Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically based rendering: From theory to implementation*. MIT Press.

[183] Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics* 2 (1993), 15–36.

[184] Adrien Poulenard and Maks Ovsjanikov. 2018. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 236:1–236:14.

[185] Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. 2019. Effective Rotation-invariant Point CNN with Spherical Harmonics kernels. *International Conference on 3D Vision (3DV)* (2019).

[186] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. PointNet: Deep learning on point sets for 3D classification and segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[187] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[188] Shi Qiu, Saeed Anwar, and Nick Barnes. 2021. Dense-Resolution Network for Point Cloud Classification and Segmentation. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2021).

[189] Shi Qiu, Saeed Anwar, and Nick Barnes. 2021. Geometric Back-projection Network for Point Cloud Classification. *arXiv preprint arXiv:1911.12885* (2021).

[190] Ravi Ramamoorthi and Pat Hanrahan. 2001. A signal-processing framework for inverse rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 117–128.

[191] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. 2018. Generating 3D faces using Convolutional Mesh Autoencoders. In *European Conference on Computer Vision (ECCV)*. 725–741.

[192] Nicolas Ray and Bruno Levy. 2003. Hierarchical least squares conformal map. In *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.* IEEE, IEEE, 263–270.

[193] Carlos Rodriguez-Pardo, Henar Dominguez-Elvira, David Pascual-Hernandez, and Elena Garces. 2023. UMat: Uncertainty-Aware Single Image High Resolution Material Capture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 5764–5774.

[194] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 234–241.

[195] Radu Alexandru Rosu and Sven Behnke. 2023. PermutoSDF: Fast Multi-View Reconstruction with Implicit Surfaces using Permutohedral Lattices. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[196] Ryan Schmidt, Cindy Grimm, and Brian Wyvill. 2006. Interactive decal compositing with discrete exponential maps. In *ACM Transactions on Graphics (ToG)*, Vol. 25. 605–613.

[197] Ryan Schmidt and Karan Singh. 2010. Meshmixer: an interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*.

[198] Stefan C. Schonsheck, Bin Dong, and Rongjie Lai. 2018. Parallel Transport Convolution: A New Tool for Convolutional Neural Networks on Manifolds. *arXiv preprint arXiv:1805.07857* (2018).

[199] Viraj Shah, Svetlana Lazebnik, and Julien Philip. 2023. JoIN: Joint GANs Inversion for Intrinsic Image Decomposition. *arXiv preprint arXiv:2305.11321* (2023).

[200] Nicholas Sharp. 2019. Polyscope. www.polyscope.run

[201] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2021. DiffusionNet: Discretization Agnostic Learning on Surfaces. *ACM Transactions on Graphics (ToG)* 41, 3 (2021), 27:1–27:16.

[202] Nicholas Sharp and Keenan Crane. 2020. A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum* 39, 5 (2020), 69–80.

[203] Nicholas Sharp, Keenan Crane, et al. 2019. geometry-central. www.geometry-central.net

[204] Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. The Vector Heat Method. *ACM Transactions on Graphics (ToG)* 38, 3 (2019).

[205] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. 2018. Mining point cloud local structures by kernel correlation and graph pooling. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[206] Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. MATch: Differentiable Material Graphs for Procedural Material Capture. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 1–15.

[207] Lin Shi, Yizhou Yu, Nathan Bell, and Wei-Wen Feng. 2006. A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graphics (ToG)* 25, 3 (2006), 1108–1117.

[208] Xiaohan Shi, Hujun Bao, and Kun Zhou. 2009. Out-of-core multigrid solver for streaming meshes. *ACM Transactions on Graphics (ToG)* 28, 5 (2009), 173.

[209] Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[210] Ayan Sinha, Jing Bai, and Karthik Ramani. 2016. Deep Learning 3D Shape Surfaces Using Geometry Images. In *European Conference on Computer Vision (ECCV)*. 223–240.

[211] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. 2012. Computational design of rubber balloons. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 835–844.

[212] Dmitriy Smirnov and Justin Solomon. 2021. HodgeNet: Learning Spectral Geometry on Triangle Meshes. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 166:1–166:11.

[213] Andrew Spielberg, Fangcheng Zhong, Konstantinos Rematas, Krishna Murthy Jatavallabhula, Cengiz Öztireli, Tzu-Mao Li, and Derek Nowrouzezahrai. 2023. Differentiable Visual Computing for Inverse Problems and Machine Learning. *arXiv preprint arXiv:2312.04574* (2023).

[214] M. Spivak. 1999. *A Comprehensive Introduction to Differential Geometry*. Number Bd. 1 in A Comprehensive Introduction to Differential Geometry. Publish or Perish, Incorporated.

[215] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7495–7504.

[216] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[217] Klaus Stüben. 2001. A review of algebraic multigrid. *J. Comput. Appl. Math.* 128, 1 (2001), 281–309.

[218] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 945–953.

[219] Cheng Sun, Guangyan Cai, Zhengqin Li, Kai Yan, Cheng Zhang, Carl Marshall, Jia-Bin Huang, Shuang Zhao, and Zhao Dong. 2023. Neural-PBIR Reconstruction of Shape, Material, and Illumination. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 18046–18056.

[220] Xiao Sun, Zhouhui Lian, and Jianguo Xiao. 2019. SRINet: Learning Strictly Rotation-Invariant

Representations for Point Cloud Classification and Segmentation. *ACM International Conference on Multimedia* (2019), 980–988.

[221] Zhiyu Sun, Ethan Rooke, Jerome Charton, Yusen He, Jia Lu, and Stephen Baek. [n.d.]. ZerNet: Convolutional Neural Networks on Arbitrary Surfaces Via Zernike Local Tangent Space Estimation. 39, 6 ([n. d.]), 204–216.

[222] Richard Sutton. 2019. The bitter lesson. *Incomplete Ideas (blog)* 13, 1 (2019), 38.

[223] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. 2018. RGCNN: Regularized graph CNN for point cloud segmentation. *ACM International Conference on Multimedia* (2018), 746–754.

[224] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2019).

[225] Nathaniel Thomas, Tess Smidt, Steven M. Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. 2018. Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds. (2018). arXiv:1802.08219

[226] T.S. Trowbridge and Karl P Reitz. 1975. Average irregularity representation of a rough surface for ray reflection. *Journal of the Optical Society of America* 65, 5 (1975), 531–536.

[227] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. 2019. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2019).

[228] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[229] Giuseppe Vecchio, Rosalie Martin, Arthur Roullier, Adrien Kaiser, Romain Rouffet, Valentin Deschaintre, and Tamy Boubekeur. 2023. ControlMat: Controlled Generative Approach to Material Capture. *arXiv preprint arXiv:2309.01700* (2023).

[230] Nitika Verma, Edmond Boyer, and Jakob Verbeek. 2018. FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2598–2606.

[231] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 125:1–125:18.

[232] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popovic. 2008. Articulated Mesh Animation from Multi-View Silhouettes. *ACM Transactions on Graphics (ToG)* 27, 3 (2008), 97.

[233] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*. 195–206.

[234] Chu Wang, Babak Samari, and Kaleem Siddiqi. 2018. Local spectral graph convolution for point set feature learning. *European Conference on Computer Vision (ECCV)* (2018).

[235] Shaodong Wang, Shuai Ma, Hui Zhao, and Wencheng Wang. 2022. A multigrid approach for generating harmonic measured foliations. *Computers & Graphics* 102 (2022), 380–389.

[236] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (ToG)* 38, 5 (2019), 146:1–146:12.

[237] Zhendong Wang, Longhua Wu, Marco Fratarcangeli, Min Tang, and Huamin Wang. 2018. Parallel Multigrid for Nonlinear Cloth Simulation. *Computer Graphics Forum* 37, 7 (2018), 131–141.

[238] Max Wardetzky. 2006. *Discrete Differential Operators on Polyhedral Surfaces - Convergence and Approximation*. Ph.D. Dissertation. Freie Universiät Berlin.

[239] Max Wardetzky, Miklós Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. 2007. Discrete quadratic curvature energies. *Computer Aided Geometric Design* 24, 8-9 (2007), 499–518.

[240] Joachim Weickert. 1998. *Anisotropic diffusion in image processing*. Vol. 1.

[241] Maurice Weiler and Gabriele Cesa. 2019. General E(2)-Equivariant Steerable CNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*. 14334–14345.

[242] Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. 2021. Coordinate Independent Convolutional Networks – Isometry and Gauge Equivariant Convolutions on Riemannian Manifolds. *arXiv preprint arXiv:2106.06020* (2021).

[243] Mark A. Wieczorek and Matthias Meschede. 2018. SHTools: Tools for Working with Spherical Harmonics. *Geochemistry, Geophysics, Geosystems* 19, 8 (2018), 2574–2592.

[244] Ruben Wiersma, Elmar Eisemann, and Klaus Hildebrandt. 2020. CNNs on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (ToG)* 4 (2020), 92:1–92:12.

[245] Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. [n.d.]. A Fast Geometric Multigrid Method for Curved Surfaces. In *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023* (2023), Erik Brunvand, Alla Sheffer, and Michael Wimmer (Eds.). ACM, 1:1–1:11.

[246] Ruben Wiersma, Ahmad Nasikun, and Elmar Eisemannand Klaus Hildebrandt. 2022. DeltaConv: Anisotropic Operators for Geometric Deep Learning on Point Clouds. *ACM Transactions on Graphics (ToG)* 41, 4 (2022).

[247] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. 2017. Harmonic networks: Deep translation and rotation equivariance. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 5028–5037.

[248] Haoqian Wu, Zhipeng Hu, Lincheng Li, Yongqiang Zhang, Changjie Fan, and Xin Yu. 2023. NeFII: Inverse Rendering for Reflectance Decomposition with Near-Field Indirect Illumination. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4295–4304.

[249] Liwen Wu, Rui Zhu, Mustafa B Yaldiz, Yinhao Zhu, Hong Cai, Janarbek Matai, Fatih Porikli, Tzu-Mao Li, Manmohan Chandraker, and Ravi Ramamoorthi. 2023. Factorized Inverse Path Tracing for Efficient and Accurate Material-Lighting Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3848–3858.

[250] Wenxuan Wu, Zhongang Qi, and Li Fuxin. 2019. PointConv: Deep Convolutional Networks on 3D Point Clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[251] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. (2019). arXiv:1901.00596

[252] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1912–1920.

[253] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[254] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. 2021. Walk in the Cloud: Learning Curves for Point Clouds Shape Analysis. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2021).

[255] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. 2021. PAConv: Position Adaptive Convolution With Dynamic Kernel Assembling on Point Clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

[256] Mutian Xu, Junhao Zhang, Zhipeng Zhou, Mingye Xu, Xiaojuan Qi, and Yu Qiao. 2021. Learning

Geometry-Disentangled Representation for Complementary Understanding of 3D Object Point Cloud. *AAAI Conference on Artificial Intelligence (AAAI)* (2021).

[257] Peiyu Xu, Sai Bangaru, Tzu-Mao Li, and Shuang Zhao. 2023. Warped-Area Reparameterization of Differential Path Integrals. *ACM Transactions on Graphics (ToG)* 42, 6 (2023), 213:1–213:18.

[258] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. 2018. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. *European Conference on Computer Vision (ECCV)* (2018).

[259] Zexiang Xu, Jannik Boll Nielsen, Jiyang Yu, Henrik Wann Jensen, and Ravi Ramamoorthi. 2016. Minimal BRDF sampling for two-shot near-field reflectance acquisition. *ACM Transactions on Graphics (ToG)* 35, 6 (2016), 1–12.

[260] Kai Yan, Christoph Lassner, Brian Budge, Zhao Dong, and Shuang Zhao. 2022. Efficient Estimation of Boundary Integrals for Path-Space Differentiable Rendering. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 123:1–123:13.

[261] Jiancheng Yang, Qiang Zhang, B Ni, L Li, J Liu, Mengdie Zhou, and Q Tian. 2019. Modeling Point Clouds With Self-Attention and Gumbel Subset Sampling. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[262] Zhangsihao Yang, Or Litany, Tolga Birdal, Srinath Sridhar, and Leonidas Guibas. 2021. Continuous geodesic convolutions for learning on 3d shapes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 134–144.

[263] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. 2016. A Scalable Active Framework for Region Annotation in 3D Shape Collections. *ACM Transactions on Graphics (ToG)* 35, 6 (2016), 210:1–210:12.

[264] Cheng Zhang, Haocheng Wan, Shengqiang Liu, Xinyi Shen, and Zizhao Wu. 2021. PVT: Point-Voxel Transformer for 3D Deep Learning. *arXiv preprint arXiv:2108.06076* (2021).

[265] Kuangen Zhang, Ming Hao, Jing Wang, Clarence W de Silva, and Chenglong Fu. 2019. Linked Dynamic Graph CNN: Learning on Point Cloud via Linking Hierarchical Features. *arXiv preprint arXiv:1904.10014* (2019).

[266] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5453–5462.

[267] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)* 40, 6 (2021), 1–18.

[268] Yingxue Zhang and Michael Rabbat. 2018. A Graph-CNN for 3D point cloud classification. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2018), 6279–6283.

[269] Youjia Zhang, Teng Xu, Junqing Yu, Yuteng Ye, Yanqing Jing, Junle Wang, Jingyi Yu, and Wei Yang. 2023. Nemf: Inverse volume rendering with neural microflake field. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 22919–22929.

[270] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. 2019. PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[271] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. 2021. Point Transformer. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2021).

[272] Yi Zheng, Kai Wei, Bin Liang, Ying Li, and Xinhui Chu. 2019. Zernike like functions on spherical cap: principle and applications in optical surface fitting and graphics rendering. *Optics Express* 27, 26 (2019), 37180.

[273] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. *arXiv preprint arXiv:1812.08434* (2018).

[274] Xilong Zhou, Miloš Hašan, Valentin Deschaintre, Paul Guerrero, Kalyan Sunkavalli, and Nima Khademi Kalantari. 2023. A Semi-Procedural Convolutional Material Prior. In *Computer Graphics Forum*. Wiley Online Library.

[275] Xilong Zhou and Nima Khademi Kalantari. 2021. Adversarial Single-Image SVBRDF Estimation with Hybrid Training. *Computer Graphics Forum* (2021).

[276] Xilong Zhou and Nima Khademi Kalantari. 2022. Look-Ahead Training with Learned Reflectance Loss for Single-Image SVBRDF Estimation. *ACM Transactions on Graphics (ToG)* 41, 6 (2022).