Gaussian Representations for Differentiable Rendering and Optimization

Leonid Keselman

CMU-RI-TR-23-70

September 2023

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Martial Hebert (Chair)
Christopher G. Atkeson
Deva Ramanan
Jon Barron, *Google*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Dedicated to my family.

Abstract

In this thesis, we explore some fast, robust and efficient methods that are generally useful across many domains. Specifically, we explore the use of Gaussian Representations in multiple application areas of computer vision and robotics.

In the first part, we provide an alternative approach to the classic hidden surface problem. In particular, we design a ray-based differentiable renderer for 3D Gaussians that can be used to solve multiple classic computer vision problems in a unified manner. For example, we can reconstruct 3D shapes from color, silhouette or optical flow, based solely on gradient-based optimization; these reconstructions are robust to input errors and reasonably fast (taking a few minutes on a laptop CPU). Similarly, we can solve for precise camera pose estimates for known objects, comparable to the quality given by classic methods. Our contributions include an alternative formulation of the hidden surface problem that sacrifices fidelity for utility, thereby obtaining fast runtimes and high-quality gradient information. We extend this renderer with differentiable optical flow and show how to export colored meshes from the reconstructions. We show examples on naturally collected videos of everyday objects.We will also cover our work on obtaining 3D Gaussian representations directly from meshes, without the need for sampling point sets.

In another line of our research, we show how Gaussian representations provide a powerful underlying representation for gradient-free optimization of classic algorithms in robotics such as stereoscopic depth matching, motion planning, visual odometry, and social navigation. We develop techniques for performing optimization based on user preferences and based on dataset variation. We show how Gaussian representations can be tuned directly from user preferences, without the need for ground-truth collection or fine-tuned metric design. Additionally, we show these optimizers can discover multiple algorithm configurations for potentially different environments, based solely on the algorithm responding to sampled configurations.

Lastly, we will touch on a novel regression-based, citation-free alternative to citation metrics for analyzing academic contributions.

Acknowledgments

"It takes a village..."

First, a great thanks must be given to my thesis advisor, Martial Hebert. Martial's humor, wit, intellectual breadth and general tolerance for creative work has greatly shaped me and this thesis. Despite his illustrious position as the Director of the Robotics Institute (when I started) and Dean of the School of Computer Science (as I write this), Martial has reliably made time to meet with me, individually, nearly every single week of my thesis. He has been patient, helpful, and listened to all comments and questions I might have throughout my time at CMU. He has never forced me to submit a deadline or work on any particular project. While his ability to engage on nearly any technical material is exceptional, he has also been the first to clearly say "I don't get it" when my presentations were unclear. Through this process, he has made me stronger technical communicator, and helped me focus on technical clarity, while strengthening my academic rigor.

Second, I have to thank Chris Atkeson, whose weekly lab meetings I've had the fortune to attend my entire time at Carnegie Mellon. Chris' charisma, charm, and far-reaching technical breadth are aspirational to students in the department, me among them. Chris introduced me to many technical areas, and always encouraged me to think outside the box of traditional academic projects. Chris' interest in real robotics, actual demonstrations, true impact, and the comfort in discarding apparent nonsense are traits I wish my thesis could even better actualize. Chris taught me that individual academic work can be thought of as art: material that shows one's world view and hopes that it can make others feel similarly, and perhaps gain some insight or inspiration from it. I've come to see all my projects as existing in this way: they are technical manifestations of a world view.

I'd like to thank the other members of my committee: Deva and Jon. While both are excellent researchers, it is their character that I've come to appreciate the most. Deva is patient, kind, and regularly willing to deep dive into technical topics with insightful questions, with a clear undercurrent of support throughout. Jon's clarity of thought, and active engagement, have made his time worth its weight in gold. His comments have been immensely valuable in helping me strengthen my work. Both of them embody being *computer vision researchers*, asking questions about understanding images, scenes, and how to represent them.

The community at The Robotics Institute has been wonderful and a pillar of my time here. I've made countless friends and acquaintances throughout my time here, and I've greatly appreciated being a part of the culture here. Without the company of the people here, I doubt this thesis would have been written. To list a few: Ankit, Ben, Thomas, Jono, Alex, Arka, Kevin, Kevin, Hunter, Martin, Neehar, Kiyn, Jason, Sudeep, Shivam, Nate, Jon, Victoria & Ben, Helen, Kayla, Simin, Maria, and many others. Adam Harley has been an excellent friend, both personally and for the frequent in-depth academic discussions. Mark Sheinin, Brian Okorn and Roberto Shu have been wonderful for their frequent office conversations. Humphrey Hu, Nick Gisolfi , Emily Simon, and Kate Shih have been wonderful friends, who I could not thank enough.

The professors at the RI are open, friendly, and engage with many of the students. For example, the three 'Matt's have always been generous with their time when crossing paths (O'Toole, Johnson-Roberson, Mason). Kris Kitani was an excellent faculty to be a TA for. I'd also like to thank Katerina, Zeynep, Henny, Oliver, Jim, David, Simon, Yannis, Srinivasa, Aaron, Sarah, and Fernando for being friendly, kind, and supportive of me and my work. Friendly administrators such as Lynetta, Nichole, Jean and Suzanne have also made me feel welcome here.

Outside the halls of CMU and the RI, this thesis would not exist if not for a long line of people who have pushed me in my life. Brian Bruno for pushing me to take a summer programming class in middle school. Carly, Katie and Alexis for teaching me to pay it forward. Kevinjeet Gill for helping me through life, including pushing me to major in Computer Science. S-K., who taught me too much about life, what it can be and how fragile it is; who truly believed I could (and should) be an academic, pushed me to do a PhD, and without whom this entire journey never would have started. Kris Pister, Ankur Mehta, and Anita Flynn, who helped me learn what research could be. Stan Melax and Sterling Orsten, for being wonderful people, and close friends, who helped teach me about computer graphics, good software, and generously helping me first appear in the peer-reviewed literature. Anders and Achin, for giving me the space to grow into my own doing computer vision. John Iselin Woodfill, for being equal parts role model, friend, and kind collaborator. Blake Lucas and Peter Henry, who taught me about 3D Computer Vision, and that you can mix fun times with hard work. Jamie Gabriel for staying late, asking good questions, and for always staying so calm. Justin Johnson, Ranjay Krishna and Ben Poole, who taught me about the high standards of being

a good teacher. Timnit for helping me think about how we think. Silvio and Fei-Fei for all the opportunities they gave me. Vincent Sitzmann, Vaggos Chatziafratis for being wonderful friends, while never shying from technical topics, and helping through tough times. Zayd Enam, Dillon Huff, Paris Syminelakis for being good friends. Ryan Julian, for being a generous friend through the years. Ermal Dreshaj, for being like family.

And lastly, I'd like to thank my parents, who helped make the person I am today.

Contents

Chapter 1

Introduction

In general, this thesis studies some fast, efficient and robust algorithms in computer vision and robotics. Of note, these methods are designed to work well in general, and experiments were constructed to validate and test their effectiveness in multiple areas. None of them were designed to maximize performance on a specific dataset or evaluation metric. Hopefully, this ensures a level of generality, flexibility and general interest to these approaches that extends past their good empirical performance on current evaluations and experimental designs. Additionally, most results in this thesis were obtained on my 2017 personal laptop, without the need for large-scale computational resources.

The bulk of this thesis is covered in several separate publications, all of which I am the first author of. My advisor is a co-author on most of them due to our weekly conversations and his helpful feedback. Curious readers may instead prefer to directly read a particular paper, which may be more neatly formatted.

- Chapter 2: Differentiable Rendering with 3D Gaussians [KH22]
- Chapter 3: Flexible Extensions to Rendering with 3D Gaussians [KH23b]
- Chapter 4: Direct Fitting of Gaussians to Meshes [KH19]
- Chapter 5: Discovering Multiple Algorithm Configurations [KH23a]
- Chapter 6: Optimizing Algorithms Based on User Preferences [Kes+23]
- Chapter 7: Learning a Citation-Free Academic Metric [Kes19]

1.1 Motivation

This thesis was motivated by a search for a *better* representation for three-dimensional objects. Early computer scientists explored many representations, including generalized cylinders [Agi72], curved patches [Coo66], and polygons [Cat72]. However, notions of what representation is *better* are usually shaped by the task at hand. Navigation for a wheeled robot [Mor83] might prefer a 2D occupancy grid [Fox01], while dense reconstruction might prefer a signed distance field [RHL02]. What representation is *better in general* may not be well defined. In this thesis, we explore how a particular representation can enable fast, robust, and simple differentiable rendering. We show how differentiable rendering can be used to solve multiple classic computer vision tasks. As such, for this work, a good representation is one which can easily produce images whose analytical gradients lead to sensible solutions.

In contrast to work from the "pattern recognition" (or "learning") literature, we are interested in a compact, interpretable representation. Statistical learning by adjusting weights has a long history, used in character recognition [Doy60; SN60; Ros60; UV61] predating early computer vision work [Rob63; Guz67; Pap66]. Neural networks have had wide success over the years, from steering vehicles [Pom88] to detecting faces [RBK98] decades before the deep learning revolution [LBH15]. Recently Neural Radiance Fields [Mil+20a] demonstrated extremely compelling results in view interpolation using MLPs to encode both density and radiance fields. Other works use MLPs to output more classic representations directly, such as signed distance fields [Par+19]. While these approaches provide effective solutions to down-stream tasks, they are not the focus here. Instead, we believe the first computer vision thesis [Rob63] adequately frames these learning methods as "better suited" for handling front-end (finding features) and back-end (making decisions) steps in a system, while techniques which "can be based on properties of three-dimensional transformations and the laws of nature" should be studied and developed, even when neural networks can adequately approximate those properties.

Our work spans three main preexisting research areas. We develop a **computer graphics** technique, designed to work with **stochastic optimization** tools, in order to solve classic **computer vision** problems. The representation should be simple, compact, and interpretable. The rendering method willingly sacrifices fidelity for

utility in optimization, focusing on smooth solutions that provide dense gradient flow. For robust vision modeling, we focus on using silhouettes and depth maps, which do not require knowledge of scene lighting and can easily be extracted by masking methods [He+17] and depth sensors [Kes+17]. Modeling colors in an image requires knowledge of lighting and material properties along with potentially high frequency information content, which isn't well supported by simple, approximate methods trying to solve coarse-grain vision problems.

On a historical note, these fields are intertwined from their very earliest days. Perhaps the first computer graphics thesis, Sketchpad [Sut63] includes optimization, error minimization and fitting of functions. The first vision thesis [Rob63] makes several mentions to pattern recognition literature while the second [Guz67] explicitly suggests parts where "standard pattern classifiers" [Nil65] may be used. Early contributors to AI [Min66] and Graphics [Sut66] both point to Robert's vision thesis as important to their fields in the same issue of Scientific American, which includes research in early 3D surface representations [Coo66]. Computer Vision research is Artificial Intelligence research, in even the earliest surveys [Fei69], which also cover the successes of statistical learning methods in playing checkers [Sam59] or building robots. Perhaps the first work in image processing [Kir+57] hinted towards using automatic pattern recognition methods to separate images into their constituent parts. Soon after, researchers deployed automated learning methods on recognition of natural images [Nag68] and written characters [Mun68]. The connections between these fields will no doubt continue into the future.

1.2 Hidden Surface Problem

Early graphics researchers studied and prioritized the *hidden surface problem*:

> The difficulty is in computing what parts of an object are visible from a given viewing position and what parts are not ... Once the hidden-surface problem is solved, shading and color are relatively easy to introduce. [Sut70]

There were many proposed solutions to this problem for early graphics hardware [Eva66; War69; Wat70; SSS74]. Of note, these early graphics researchers also

developed meshes, the popularity of which may have originated from the ease at which early digitized objects were painted over and turned into polygons [Gab15]. [Gab15] studies the history of the hidden surface problem and how early researchers tackled it.

A main contribution of this thesis is to provide an alternative solution to the hidden surface problem. Most rendering approaches are typically not differentiable, and we aim to formulate an alternative approach that enables robust gradient flow. Classic rendering approaches are discontinuous for two primary reasons:

- Rays either intersect known geometry or they do not. If they do not intersection any geometry, there is no gradient information to the shape.
- Rays that intersect geometry typically only intersect the front-most component of geometry. This limits gradient flow to only a single piece of geometry.

Instead, our rendering equations in Chapters 2 and 3 have continuous, differentiable alternatives to the two steps above.

- Shapes are represented with mixtures 3D Gaussians, which have infinite support and are non-zero everywhere. This implies that all rays intersect all geometry in the scene to some degree.
- Final intersection values are combined as a weighted average of positive weights, leading to some contribution from all geometric components.

Final intersections can be computed with either a novel sort-free approach (Sections 2.3 and 3.3.2) or with typical alpha-compositing (Section 3.3.3). These choices enable differentiable rendering that is smooth due to both the underlying geometry (whose intersections are defined via maximization of some quantity), and due to the smooth computation of final intersection locations. This aids smooth optimization, and export to traditional discontinuous representations requires additional steps (Sections 2.15 and 3.6).

Chapter 2

Fuzzy Metaballs: 3D Gaussians for Differentiable Rendering

This chapter introduces an approach to differentiable rendering of shapes where the underlying shape is represented as a mixture of 3D Gaussians. It is largely similar to our publication on this topic [KH22].

Originally, 3D Gaussians in the rendering and graphics literature were called *Metaballs* (or atoms, soft objects, blobs, blobbies). The original approach was used to render visualizations of atoms, and defined the surface at a specific isosurface level [Bli82]. Instead, we use 3D Gaussians as objects that are defined everywhere, a property that enables robust optimization for differentiable rendering applications such as pose estimation and shape reconstruction. Hence we call the approach *Fuzzy* Metaballs, to note that we're extending metaballs and also using them as infinite support objects without a specific isosurface value.

As an alternative to searching for a discrete isosurface level, we instead intersect each Gaussian at a maximal value of some objective. We derive three alternative rules for maximal intersection of Gaussians, which correspond to different orders of polynomial: linear, quadratic, and cubic. Along with the title of [Bli82], this is why we sometimes refer to these as algebraic surfaces of different order [Zar35]. This approach to rendering provides smooth interpolation between the objects, leading to surface interpolation between individual components, which behave more like flat planes (especially under the linear formulation) (see Figs. 2.1 and 2.3) than ellipsoids.

An interesting part of this chapter is studying parameter efficient representations of 3D shapes. We consider the scaling of different representations in Section 2.6. In general, we focus on having under-defined representations which can be robust to errors in annotations (Section 2.10) and only model as much shape complexity as is needed for coarse shape representation for potential robotics applications like planning, manipulation, and collusion avoidance.

In Chapter 3 we simplify both the models used and some of the presentation of similar material, so interested readers may benefit from the summary in Section 3.3.1.

2.1 Introduction

Rendering can be seen as the inverse of computer vision: turning 3D scene descriptions into plausible images. There are countless classic rendering methods, spanning from the extremely fast (as used in video games) to the extremely realistic (as used in film and animation). Common to all of these methods is that the rendering process for opaque objects is discontinuous; rays that hit no objects have no relationship to scene geometry and when intersections do occur, they typically only interact with the front-most component of geometry.

Differentiable Rendering is a recent development, designing techniques (often sub-gradients) that enable a more direct mathematical relationship between an image and the scene or camera parameters that generated it. The easy access to derivatives allows for statistical optimization and natural integration with gradient-based learning techniques. There exist several recent differentiable renderers which produce images comparable in fidelity to classic, non-differentiable, photorealistic rendering methods [Lai+20; Mil+20a; Nim+19; Zha+20].

This chapter presents a different approach: a differentiable renderer focused on utility for computer vision tasks. We are interested in the quality and computability of gradients, not on matching the exact image formation task. We willingly sacrifice fidelity for computational simplicity (and hence speed). Our method focuses on a rendering-like process for shapes which generates good gradients that rapidly lead to viable solutions for classic vision problems. Other methods may produce more pleasing images, but we care about the quality of our local minima and our ability to easily find those minima. Our experiments show how, compared to classic methods,

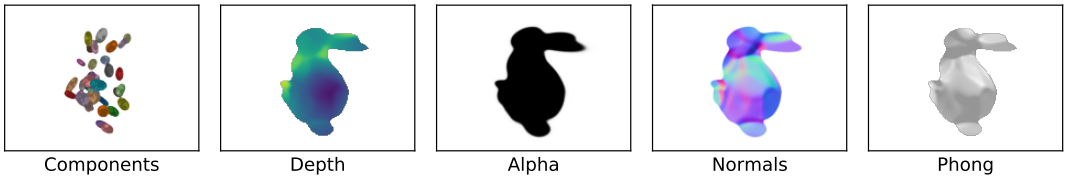


Figure 2.1: **Our differentiable renderer producing images of Stanford bunny**, using a representation with 400 parameters. From left to right: the 40 components at one standard deviation, followed by our differentiable renderer generating depth, alpha, surface normals and a shaded image.

differentiable renderers can be used to solve classic vision problems using only gradient descent, enabling a high degree of robustness to noise such as under-segmented masks or depth sensor artifacts.

Our approach is built on a specific 3D representation. Existing representations often have undesirable properties for rendering or optimization. Point clouds require splatting or calculating precise point sizes [Yif+19]. Meshes explicitly represent the object surface, making changes of genus difficult. Other representations require optimization or numerical estimation of ray-shape intersections [Bli82; Mil+20a]. Our proposed method is formulated with independent rays, represents object surfaces implicitly and computes ray termination in closed form.

Most existing differentiable renders focus on GPU performance. However, GPUs are not always available. Many robotics platforms do not have a GPU [Tom+12] or find it occupied running object detection [YS19], optical flow [TD20] or a SLAM method [Mil+20b]. While a single method may claim to be real-time on a dedicated GPU [TD21a], an autonomous system requires a sharing of resources. To run in parallel with the countless GPU-friendly techniques of today, CPU-friendly methods are desirable. Thus, while our method is implemented in JAX [Bra+18], supporting CPU and GPU backends, our focus is typically on CPU runtimes.

Lastly, in the era of deep learning, techniques which support gradient-based optimization are desirable. Since our objects have an explicit algebraic form, gradients are simple and easy to compute. Importantly, every pixel has a non-zero (if very slight) relationship with each piece of geometry in the scene (even those behind the camera!). This allows for gradient flow (up to machine precision), even when objects start far from their initialization. While this can also true of large over-parmaterized

implicit surfaces (such as NeRF [Mil+20a]), our representation is extremely compact and each parameter has approximate geometric meaning.

2.2 Related Work

Early work in 3D shape representation focused on building volumes from partial observations [Agi72] but most modern methods instead focus on surface representation. Meshes, point clouds and surfels [Pfi+00] focus on representing the exterior of an object. In contrast, our method works by representing volumes, and obtaining surface samples is implicit; similar to recent work on implicit neural surfaces [Mil+20a]. Tetrahedral volumetric representations have also been proven useful for geometric processing [Hu+18].

In using low-fidelity representations, our work is hardly unique. Often learning-based methods settle for pseudorendering [LKL18] or even treating images as layers of planar objects [TS20]. Settling for low fidelity contrasts sharply with a wide array of differentiable renderers focused on accurate light transport, which are slower but can simulate subtle phenomena [BLD20; Zha+20]. High-quality results can also be obtained by using learning methods and dense voxel grids [Lom+19]. In psychology, it has been shown that people can be primed to use a recognition pathway that focuses on low-fidelity, blobby representations [SO94].

Differentiable Rendering has many recent works. OpenDR [LB14] demonstrated pose updates for meshes representing humans. Neural Mesh Renderer [KUH17] developed approximate gradients and used a differentiable renderer for a wide array of tasks. SoftRasterizer [Liu+19] developed a subgradient function for meshes with greatly improved gradient quality. Modular Primitives [Lai+20] demonstrated fast, GPU-based differentiable rendering for meshes with texture mapping. Differentiable Surface Splatting [Yif+19] developed a differentiable renderer for point clouds by building upon existing rendering techniques [Zwi+01]. Conversion of point clouds to volumes is also differentiable [ID18]. Pulsar [LZ21] uses spheres as the primary primitive and focuses on GPU performance. PyTorch3D [Rav+20] implements several of these techniques for mesh and point cloud rendering. Some methods exploit sampling to be generic across object representation [Col+21]. Many methods integrate with neural networks for specific tasks, such as obtaining better descriptors [Li+20]

or predicting 3D object shape from a single images [Che+19; Tew+17].

The use of an algebraic surface representation, which came to be known as *metaballs* can be attributed to Blinn [Bli82]. These algebraic representations were well studied in the 1980s and 1990s. These include the development of ray-tracing approximations [Hec86; WMW86; WT90] and building metaball representations of depth images [Mur91]. Non-differentiable rendering metaballs has many methods, involving splatting [ALD06], data structures [Gou+10; SI12] or even a neural network [Hor19].

Metaballs, especially in our treatment of them, are related to the use of Gaussian Mixture Models (GMMs) for surface representation. Our method could be considered a *differentiable renderer for GMMs*. Gaussian Mixture Models as a shape representation has some appeal to roboticists [OTM19; TOM18]. Methods developed to render GMMs include search-based methods [SM20] and projection for occupancy maps [OTM19]. Projection methods for GMMs have also found application in robot pose estimation [Hua+20]. In the vision community, GMMs have been studied as a shape representation [Eck+16] and used for pose estimation [EKK18; Eck+15]. In the visual learning space, GMMs [Her+20], or their approximations [Gen+20] have also been used.

Concurrent work also uses Gaussians for rendering. VoGE [Wan+23] uses existing volume rendering techniques [Max95; Mil+20a]. Others use a DGT to build screen-space Gaussians for point clouds [MWK22]. In contrast, our contribution is the development of an approximate differentiable renderer that produces fast & robust results.

2.3 Fuzzy Metaballs

Our proposed object representation, dubbed Fuzzy Metaballs, is an algebraic, implicit surface representation. Implicit surfaces are an object representations where the surface is represented as

$$F(x, y, z) = 0. \tag{2.1}$$

While some methods parameterize F with neural networks [Mil+20a], Blinn's

algebraic surfaces [Bli82], also known as blobby models or metaballs, are defined by

$$F(x, y, z) = \sum_i \lambda_i P(x, y, z) - T, \tag{2.2}$$

where $P(x, y, z)$ is some geometric component and T is a threshold for sufficient density. While Blinn used isotropic Gaussians (hence balls), in our case, we use general multidimensional Gaussians that form ellipsoids:

$$P(\vec{x}) = |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\vec{x} - \mu)^T \Sigma^{-1} (\vec{x} - \mu)\right). \tag{2.3}$$

In contrast to classic metaballs, we relax the restriction on T being a hard threshold set by the user and instead develop a ray-tracing formulation for Gaussians which implicitly defines the surface; hence *fuzzy* metaballs. To achieve this, we develop two components: a way of defining intersections between Gaussians and rays (Section 2.4.1), and a way of combining intersections across all Gaussians (Section 2.4.2). In our definition, all rays always intersect all Gaussians, leading to smooth gradients. The *fuzzy* surface locations are not viewpoint invariant.

Our implementation is in JAX [Bra+18], enabling CPU and GPU acceleration as well as automatic backpropogation. The rendering function that takes camera pose, camera rays and geometry is 60 lines of code. To enable constraint-free backpropogation, we parameterize Σ^{-1} with its Cholesky decomposition: a lower triangular matrix with positive diagonal components. We ensure that the diagonal elements are positive and at least 10^{-6}. The determinant is directly computed from a product of the diagonal of L. When analyzing ray intersections, one can omit the $|\Sigma|^{-\frac{1}{2}}$ term as maximizing requires only the quadratic form. For example, $\vec{x}$ is replaced with a ray intersection of $\vec{v}t$ with $\vec{v} \in \mathbf{R}^3$ and $t \in \mathbf{R}$:

$$s(vt) = (vt - \mu)^T \Sigma^{-1} (vt - \mu), \tag{2.4}$$

giving a Mahlanobis distnance [Mah36] that is invariant to object scale and allows us to use constant hyperparameters, irrespective of object distance. Using probabilities would be scale-sensitive as equivalent Gaussians that are further are also larger and would have smaller likelihoods at the same points.

To produce an alpha-mask, we simply have two hyperparameters for scale and offset and use a standard sigmoid function:

$$\alpha = \sigma\left(\beta_4\left[\sum_i \lambda_i \exp(-\frac{1}{2}s(vt))\right] + \beta_5\right). \tag{2.5}$$

2.4 Approximate Differentiable Rendering

Instead of using existing rendering methods, we develop an approximate renderer that produces smooth, high-quality gradients. While inexact, our formulation enables fast and robust differentiable rendering usable in an analysis by synthesis pipeline [Bel+61]. We split the process into two steps: intersecting each component independently in Section 2.4.1 and combining those results smoothly in Section 2.4.2.

2.4.1 Intersecting Gaussians

What does it mean to have a particular intersection of a ray with a Gaussian? We propose three methods. The *linear* method is where the ray intersects the Gaussian at the point of highest probability. Maximizing Eq. (2.4) is solved by

$$t = \frac{\mu^T\Sigma^{-1}v}{v^T\Sigma^{-1}v}. \tag{2.6}$$

An alternative view is a volume model, intersecting at the maximum magnitude of the gradient of the Gaussian:

$$||\nabla p(tv)||^2 = P(tv)^2(tv-\mu)^T\Sigma^{-1}\Sigma^{-1}(tv-\mu). \tag{2.7}$$

Obtaining the gradient of Eq. (2.7) and setting it equal to zero leads to a cubic equation, hence the *cubic* method. Defining $m = \Sigma\mu$ and $r = \Sigma v$ leads to:

$$\begin{aligned} 0 = &-t^3(r^Tr)(v^Tr) \\ &+t^2\left[(m^Tr + r^Tm)(v^Tr) + (r^Tr)(v^Tm)\right] \\ &-t\left[(m^Tm)(v^Tr) + (m^Tr + r^Tm)(v^Tm) - r^Tr\right] \\ &+(m^Tm)(v^Tm) - r^Tm. \end{aligned}$$

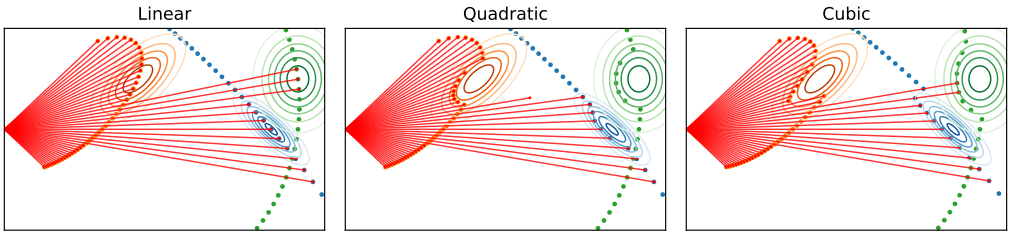


Figure 2.2: Two dimensional version of our approximate renderer with camera rays cast from the center left. Three components are shown by their contour maps and their intersections with dots. The blended results are shown with red rays.

While standard formulas exist for the cubic, the higher order polynomial all-but-ensures that numerical issues will arise. We implement a numerically stable solver for the cubic [Bli07]. However, even the numerically stable version produces problematic pixels in 32bit floating point. Errors at a rate of about 1 in 1,000 produce NaNs and make backpropagation impossible.

The *quadratic* method approximates the cubic by intersecting the Gaussian at the one standard deviation ellipsoid. Clipping the inside of square roots to be non-negative leads to reasonable results when the ray misses the ellipsoid.

$$t^2 v^T \Sigma^{-1} v - 2tv^T \Sigma^{-1} \mu + \mu^T \Sigma^{-1} \mu = 1$$

$$a = v^T \Sigma^{-1} v \qquad b = -2v^T \Sigma^{-1} \mu \qquad c = \mu^T \Sigma^{-1} \mu - 1$$

Figures 2.2 and 2.3 illustrate all three methods. The linear method produces smooth surfaces and the quadratic surface shows the individual ellipsoids protruding from the surface of the object and the cubic shows artifacts.

In 3D evaluation on objects, for a forward pass, the linear method is the fastest, the quadratic method takes 50% longer and the cubic method takes twice as long as the linear method. The quadratic method has the lowest errors in depth and mask errors. However, due to its stability, in all evaluation outside this section, we use the linear method.

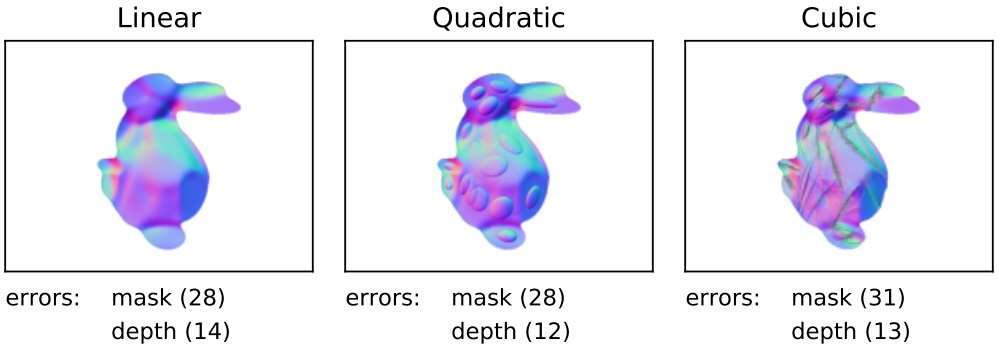


Figure 2.3: Visual examples of normal maps from different methods of ray intersection, along with the respective mask and depth errors. See Section 2.4.1 for details

2.4.2 Blending intersections

We present a particular solution to the hidden-surface problem [SSS74]. Our method is related to prior work on *Order Independent Transparency (OIT)* [End+10; MB13] but extended to 3D objects with opaque surfaces. We combine each pixel's ray-Gaussian intersections with a weighted average

$$t_f = \frac{1}{\sum_i w_i} \sum_i w_i t_i. \tag{2.8}$$

The weights are an exponential function with two hyperparameters β_1 and β_2 balancing high-quality hits versus hits closer closer to the camera:

$$w_i = \exp\left(\beta_1 s(vt_i) h(t_i) - \frac{\beta_2}{\eta} t_i\right). \tag{2.9}$$

We include a term (η) for the rough scale of the object. This, along with use of Eq. (2.4) allows our rendering to be invariant to object scale. We also include an extra term to down-weight results of intersections behind the camera with a simple sigmoid function:

$$h(t) = \sigma\left(\frac{\beta_3}{\eta} t\right). \tag{2.10}$$

Our blending solution requires only $O(N)$ evaluations of Gaussians for each ray.

2.4.3 Obtaining Fuzzy Metaballs

A representation can be limited in utility by how easily one can convert to it. We propose that, unlike classic Metaballs, Fuzzy Metaballs have reasonably straightforward methods for conversion from other formats.

Since we've developed a differentiable renderer, one can optimize a Fuzzy Metaball representation from a set of images. One could use several different losses, but experiments with silhouettes are described in Section 2.7.2.

If one has a mesh, the mathematical relationship of Fuzzy Metaballs and Gaussian Mixture Models can be exploited by fitting a GMM with Expectation-Maximization [DLR77]. With Fuzzy Metaballs being between a surface and volume representation, there are two forms of GMM one could fit. The first is a surface GMM (sGMM) as used by many authors [Eck+16; KH19; TOM18], where a GMM is fit to points sampled from the surface of the object. The second is to build a volumetric GMM (vGMM). To build a vGMM, one takes a watertight mesh [KBH06], and samples points from the interior of the object. Fitting a GMM to these interior points is what we call a volumetric GMM. Both representations can then further be optimized using the differentiable renderer. Our experiments show that both forms of GMM initialization work well, but we use vGMMs in our experiments.

Extraction is also straightforward. Point clouds can easily be sampled from our proper probability distributions. Extracting a mesh is possible by running marching cubes [LC87] with an optimized iso-surface level. Details in Section 2.15.

2.5 Data

We use ten models for evaluation: five from the Stanford Model Repository [Lev+05] (arma, buddha, dragon, lucy, bunny), three from Thingi10K [ZJ16] (gear, eiffel, rebel) and two from prior rendering literature (yoga, plane). All ten are used for reconstruction, and seven are used for pose estimation. We selected objects with different scales, genus, and variety in features. We choose 40 component FMs based on prior literature suggesting 20 to 60 GMMs for object representation [EKK18].

Table 2.1: **Runtimes in milliseconds** with $\mu \pm \sigma$ for rendering images and performing gradient updates in pose estimation with comparable fidelity (Section 2.6). CPU performance may be a fairer comparison as our method is 60 lines of JAX [Bra+18] code and lacks a custom CUDA kernel. CUDA numbers use 160 x 120 images on a Quadro P2000, while CPU use 80 x 60 images on an i5-7287U.

Method	**Forward CUDA**	**Backwards CUDA**	**Forward CPU**	**Backwards CPU**
Point Cloud [Rav+20]	12.1 ± 0.5	23.4 ± 0.5	18.0 ± 1.0	23.8 ± 4.0
Pulsar [LZ21]	7.8 ± 0.3	11.2 ± 0.4	16.4 ± 1.4	63.6 ± 7.9
SoftRas Mesh [Liu+19; Rav+20]	17.0 ± 0.4	27.2 ± 0.5	21.5 ± 2.0	384.7 ± 93.8
Fuzzy Metaballs	$\mathbf{3.0 \pm 0.2}$	$\mathbf{9.6 \pm 0.5}$	$\mathbf{3.0 \pm 0.15}$	$\mathbf{13.2 \pm 1.4}$

2.6 Comparing Representations

Fairly comparing object representations requires some notion of what to hold constant. As the parameter counts of each representation increase, so do their representational ability. It would be unfair to compare a million point point-cloud against a 100 face triangle mesh. Since our goal is utility in vision tasks, our definition of fidelity will also be task-centric.

In this case, our metric of fidelity will be a representation's *perturbation sensitivity*. We define this as the pose error obtained when optimizing an object's camera pose given a depth map, when the optimization process was initialized with ground truth camera pose. The given depth map is of the full representation object, but the methods are evaluated using lower fidelity versions, leading to perturbations of optimal pose and our fidelity metric. Pose errors are reported using the geometric mean of rotation error and translation error.

Results of our fidelity experiments can be seen in Fig. 2.4. We evaluate point clouds and meshes using a standard Iterative Closest Point (ICP) method [ZPK18], with the point clouds randomly subsampled and the meshes undergoing decimation [GH97]. We also use PyTorch3D [Rav+20], a differential mesh renderer, and obtain its perturbation curve. These experiments are conditional on an experimental setup and methods used, and thus these results may change under different conditions.

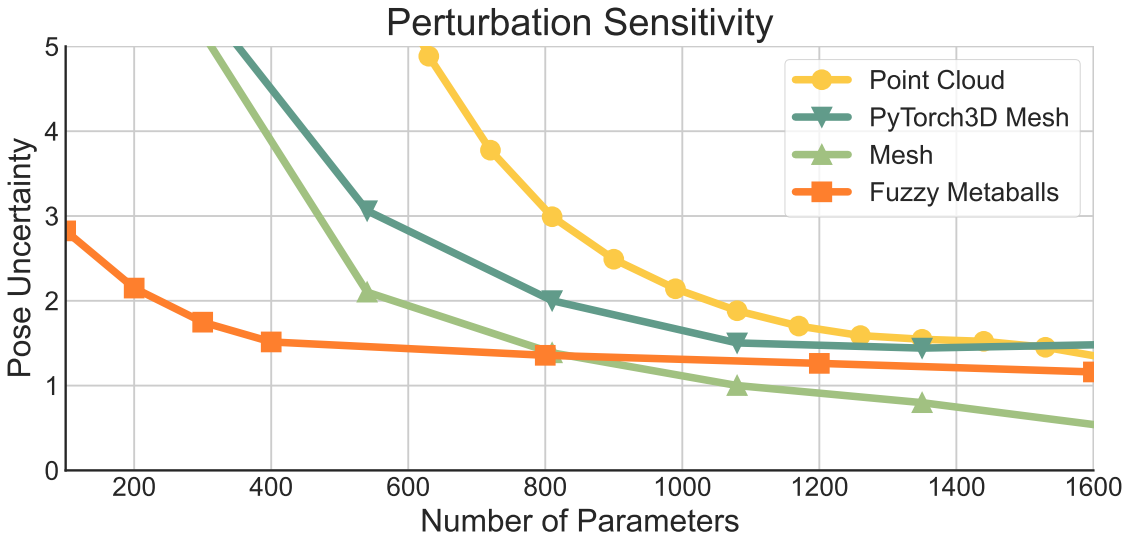


Figure 2.4: **Perturbation sensitivity** is the average error in pose when registration is performed with ground truth pose as initialization. See Section 2.6 for details. The underlying ground truth is a decimated mesh, so only the mesh representation approaches exactly zero error while other asymptote at a higher mark.

In our experiments, a 40 component Fuzzy Metaball (the size we throughout across this chapter) produces a pose uncertainty equivalent to a 470 point point cloud (roughly triple the parameter count of a fuzzy metaball) and 85 vertex, 170 triangle mesh (roughly twice the parameter count). These are the sizes use throughout the rest of the chapter, in our attempt to keep comparisons fair.

2.7 Experiments

For comparison against other Differentiable Renderers, we use the methods implemented in PyTorch3D [Rav+20], which has a wide variety of techniques with well-optimized routines. The mesh rendering method is an implementation of the *SoftRasterizer* [Liu+19]. For point clouds, PyTorch3D cites *Direct Surface Splatting* [Yif+19], while also implementing the recent *Pulsar* [LZ21].

With the fidelity of different object representations normalized out (Section 2.6), we can compare the runtime performance in a fair way, with times shown in in Table 2.1. On the CPU, where comparisons are more equal (due to lacking a custom CUDA

kernel), our renderer is 5 times faster for a forward pass, and significantly faster (30x) for a backwards pass compared to the mesh rendering methods. The point cloud renderer is more comparable in runtime to ours but need a pre-specified point size, often producing images with lots of holes (when points are too small) or a poor silhouette (when points are too big).

To the demonstrate the ability our differentiable renderer to solve classic computer vision tasks, we look at pose estimation (Section 2.7.1) and 3D reconstruction from silhouettes (Section 2.7.2). Our renderer is a function that takes camera pose and geometry, and produces images. It seems natural to take images and evaluate how well either camera pose or geometry can be reconstructed, when the other is given. All five hyperparameters for our rendering algorithm ($\beta_{1,2,3,4,5}$) were held constant throughout all experiments.

Since pose estimation and shape from silhouette (SFS) are classic computer vision problems, there are countless methods for both tasks. We do not claim to be the best solution to these problems, as there are many methods specifically designed for these tasks under a variety of conditions. Instead, we seek to demonstrate how our approximate differentiable renderer is comparable in quality to typical solutions, using only gradient descent, without any regularization.

2.7.1 Pose Estimation

Many differential renderers show qualitative results of pose estimation [Liu+19; Yif+19]. We instead perform quantitative results over our library of models rendered from random viewpoints. Methods are given a perturbed camera pose ($\pm 45°$ rotation and a random translation up to 50% of model scale) and the ground truth depth image from the original pose. The methods are evaluated by their ability to recover the original pose from minimizing image-based errors. The resulting pose is evaluated for rotation error and translation error. We quantify the score for a model as the geometric mean of the two errors. All methods are tested on the same random viewpoints and with the same random perturbations.

For Fuzzy Metaballs, we establish projective correspondence [RL01] and optimize

Table 2.2: **Pose Estimation Results**. Pose Errors are reported with a geometric mean of rotation and translation error. The reported numbers are mean ± IQR. We report results clean data and data with simulated depth and silhouette noise.

		Parameters	**Noise-Free Error**	**Noisy Error**
Initialization			20.2 ± 18	20.2 ± 18
Pulsar [LZ21]		1,200	20.2 ± 18	20.2 ± 18
Point Cloud [Rav+20]		1,200	18.5 ± 16	18.4 ± 16
SoftRas Mesh [Liu+19]		750	14.9 ± 15	17.0 ± 17
Equal Fidelity (Plane) [ZPK18]	ICP	1,200	10.8 ± 12	8.2 ± 3.3
Equal Fidelity (Point) [ZPK18]	ICP	1,200	7.6 ± 9.9	8.7 ± 6.6
High Fidelity (Plane) [ZPK18]	ICP	120,000	8.2 ± 0.8	8.0 ± 3.6
High Fidelity (Point) [ZPK18]	ICP	120,000	6.2 ± 3.7	6.8 ± 3.3
Fuzzy Metaballs		400	**4.0 ± 1.5**	**4.2 ± 2.1**

silhouette cross-entropy loss averaged over all pixels:

$$CE(\alpha, \hat{\alpha}) = \alpha \log(\hat{\alpha}) + (1 - \alpha) \log(1 - \hat{\alpha}). \tag{2.11}$$

Estimated alpha is clipped to $[10^{-6}, 1 - 10^{-6}]$ to avoid infinite error. We also evaluate with an additional depth loss of $MSE(z, \hat{z})$ where $|z|$ normalizes the errors to be invariant to object scale and comparable in magnitude to $CE(\alpha, \hat{\alpha})$.

$$MSE(z, \hat{z}) = \left|\left| \frac{(z - \hat{z})}{|z|} \right|\right|_2 \tag{2.12}$$

There is a subtle caveat in the gradients of Fuzzy Metaballs. The gradient of the translation scales by the inverse of model scale.. We correct for this by scaling the gradients by η^2. Alternatively one could scale the input data to always be of some canonical scale [YLJ13]. To maintain scale invariance, we limit our use of adaptive learning rate methods to SGD with Momentum.

We provide point cloud ICP results for point-to-point and point-to-plane meth-

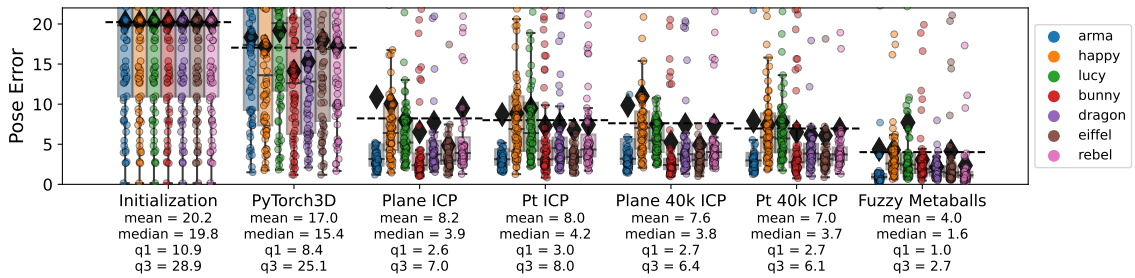


Figure 2.5: **Noisy Pose Estimation** Dashed lines are averages for the method, while the black diamonds show the average for that method and model. Here Fuzzy Metaballs win in all statistical measures, typically by a factor of ≈ 2.

ods [RL01] as implemented by Open3D [ZPK18]. For the differentiable rendering experiments, we use PyTorch3D [Rav+20] and tune its settings (Section 2.14). All differentiable rendering methods use the same loss, learning rate decay criteria and are run until the loss stops reliably decreasing.

Pose Estimation Results

Overall results are found in Table 2.2 and a more detailed breakdown in Fig. 2.5. All methods sometimes struggle to find the correct local minima in this testing setup. Prior differentiable renderers significantly under-performed classic baselines like ICP, while our approximate renderer even outperforms the ICP baselines under realistic settings with synthetic noise.

ICP on noise-free data had bimodal results: it typically either recovered the correct pose to near machine precision or it fell into the wrong local minima. Despite having a higher mean error, ICP's median errors on noise-free data were $\frac{1}{10}$ of Fuzzy Metaballs (FMs). With noisy data, this bimodal distribution disappears and Fuzzy Metaballs outperform on all tested statistical measures. FMs even outperformed ICP with high-fidelity point clouds, suggesting a difference in method not just fidelity. This may be due to our inclusion of a silhouette loss, the benefits of projective correspondence over the nearest neighbors used by this ICP variant [ZPK18] or the strengths of visual loss over geometric loss [Tri+00].

2.7.2 3D Reconstruction

Reconstruction experiments are common in the differential rendering literature [LZ21; Yif+19]. However, instead of optimizing with annotations of color [Lai+20] or normals [Yif+19], we instead only optimize only over silhouettes, as in the classic Shape From Silhouette (SFS) [CBK05]. Unlike many prior examples in the literature, which require fine-tuning of several regularization losses [Rav+20; Yif+19], we use no regularization in our experiments and can keep constant settings for all objects.

We initialize with a sphere (isosphere for meshes, an isotropic Gaussian of points for point clouds and a small blobby sphere for Fuzzy Metaballs). Given a set of silhouette images and their camera poses, we then optimize silhouette loss for the object. In our experiments, we use 64 x 64 pixel images and have 32 views. For these experiments, we resize all models to a canonical scale and use the Adam [KB15] optimizer. For baseline hyperparameters, we use the PyTorch3D settings with minimal modification. For SoftRas, we use a twice subdivided icosphere. For NeRF [Mil+20a], we use a two layer MLP with 30 harmonic function embedding with 128 hidden dimension and the same early exit strategy as FMs.

Inspired by artifacts seen in real videos (Fig. 2.9), we produce a noisy silhouette dataset where training data had $\frac{1}{8}$ of each silhouette under-segmented (Fig. 2.8) in 16 of 32 images by clustering silhouette coordinates [Scu10a] and removing a cluster.

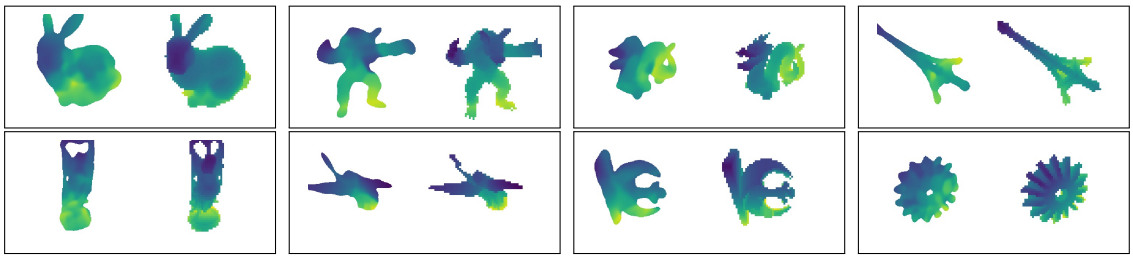


Figure 2.6: **Shape from Silhouette (SFS)** reconstructions. On the left is a 40 component Fuzzy Metaball result and the right is the mesh ground-truth of about 2,500 faces, both colored by depth maps.

Shape From Silhouette Results

We show qualitative reconstructions from Fuzzy Metaballs (Fig. 2.6), along with quantitative results against baselines (Table 2.3) and some example reconstructions

Table 2.3: **Shape from Silhouette reconstruction fidelity** as measured by cross-entropy silhouette loss on 32 novel viewpoints for each of 10 sample models. Runtimes were the average per model and performed on CPU. Results show $\mu \pm \sigma$.

	Time (s)	Noise-Free Recon. Error	Noisy Recon. Error
Voxel Carving [MA83; ZPK18]	82	0.31 ± 0.100	1.119 ± 0.367
PyTorch3D Point [Rav+20]	185	0.075 ± 0.066	0.100 ± 0.079
PyTorch3D Mesh [Liu+19]	3008	0.062 ± 0.049	0.072 ± 0.051
NeRF [Mil+20a]	7406	**0.032 ± 0.022**	0.062 ± 0.063
Fuzzy Metaballs	**68**	0.040 ± 0.015	**0.055 ± 0.016**

from all methods (Fig. 2.8).

Overall, we found that our method was significantly faster than the other differentiable renderers, while producing the best results in the case of noisy reconstructions. Classic voxel carving [MA83] with a 384^3 volume was reasonably fast, but the 32 views of low resolution images didn't produce extremely sharp contours (Fig. 2.24). With under-segmentation noise, voxel carving fails completely while the differentiable renderers reasonably reconstruct all models.

Among the differentiable renders, we can see how the mesh-based approach struggles to change genus from a sphere to the Eiffel tower. The point cloud renderer lacks the correct gradients to successfully pull spurious points into the model. NeRF [Mil+20a] performs reasonably well in shape from silhouette, even with spurious masks. In fact, it was the best performer for noise-free data, and in a majority of the reconstructions in noisy data (its mean performance was hurt by results on `eiffel` and `lucy` with long thin surfaces). NeRF is a sophisticated model with many settings, and it may have a configuration where it successfully reconstructs all the models, but due to its dense volumetric rendering and use of an MLP, it is 100x slower than our low degree of freedom representation.

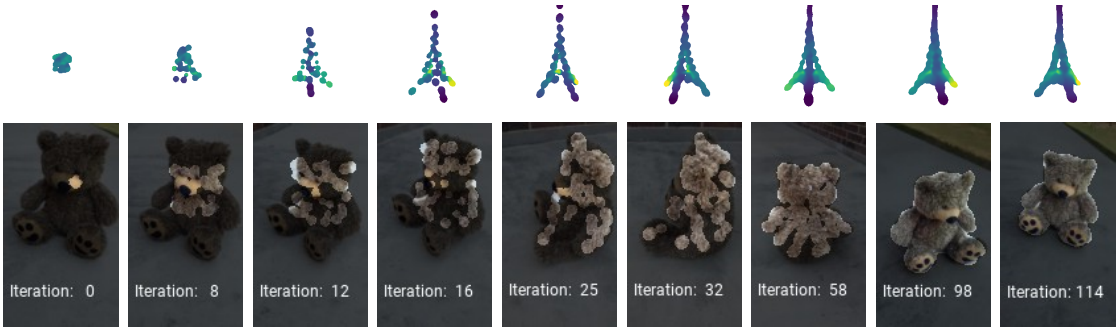


Figure 2.7: **Shape from Silhouette steps** Top row shows synthetic data with reconstructed depth. Bottom row shows reconstructed masks for a CO3D video [Rei+21].

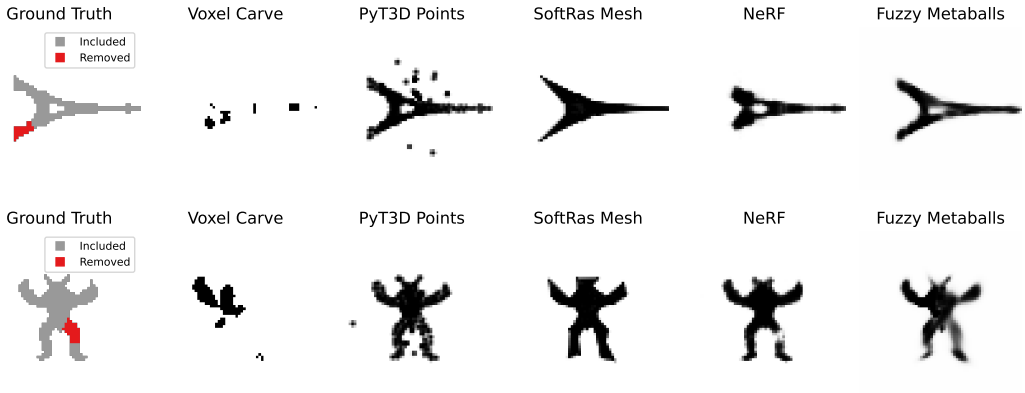


Figure 2.8: **Shape from Silhouette Results** with simulated under-segmentation.

2.8 Discussion

The focus of our approximate differentiable rendering method has been on shape. While it is possible to add per-component colors to Fuzzy Metaballs (Fig. 2.20), that has not been the focus of our experiments. Focusing on shape allows us to circumvent modeling high-frequency color textures, as well as ignoring lighting computations. This shape-based approach can use data from modern segmentation methods [He+17] and depth sensors [Kes+17]. Low-degree of freedom models have a natural robustness and implicit regularization that allows for recovery from significant artifacts present in real systems. For example, Fig. 2.9 shows robust recovery from real over/under-segmentation artifacts in video sequences.

Our approximate approach to rendering by using OIT-like methods creates a

trade-off. The downside is that small artifacts can be observed since the method coarsely approximates correct image formation. The benefits are good gradients, speed & robustness, all of which produce utility in vision tasks.

Compared to prior work [LZ21; Liu+19], our results do not focus on the same areas of differentiable rendering. Unlike other work, we do not perform GPU-centric optimizations [Lai+20]. Additionally, prior work focuses on producing high-fidelity color images (and using them for optimization). Unlike prior work, we benchmark our method across a family of objects and report quantitative results against classic baselines. Unlike some popular implicit surface methods such as the NeRF [Mil+20a] family, our object representation is low degree of freedom, quick to optimize from scratch, and all the parameters are interpretable with geometric meaning.

While our experiments focus on classic computer vision tasks such as pose estimation or shape from silhouette, the value of efficiently rendering interpretable, low degree of freedom models may have the biggest impact outside of classic computer vision contexts. For example, in scientific imaging it is often impossible to obtain high-quality observations since the sensors are limited. For example, in non-light-of-sight imaging [TSG19], sonar reconstruction [WGK20], lightcurve inversion [KT01] and CryoEM [BPF15; Zho+21]. In all these contexts, getting good imaging information is extremely hard and low degree of freedom models could be desirable.

2.9 Conclusion

Approximate differentiable rendering with algebraic surfaces enables fast analysis-by-synthesis pipelines for vision tasks which focus on shapes, such as pose estimation and shape from silhouette. For both tasks, we show results with realistic, simulated noise. The robustness of our approach enables it to runs naturally on silhouettes extracted from real video sequences without any regularization. Whereas classic methods can struggle once noise is introduced, differentiable renderers naturally recovery by using stochastic optimization techniques. By using gradient-based optimization, differentiable rendering techniques provide a robust solution to classic vision problems. Fuzzy Metaballs can enable low-latency differential rendering on CPUs. Our formulation connects algebraic surfaces [Bli82] used in graphics with Gaussian Mixture Models [Eck+16] used in vision. These provide a compact, interpretable

representation for shape with many uses.

2.10 Video Results

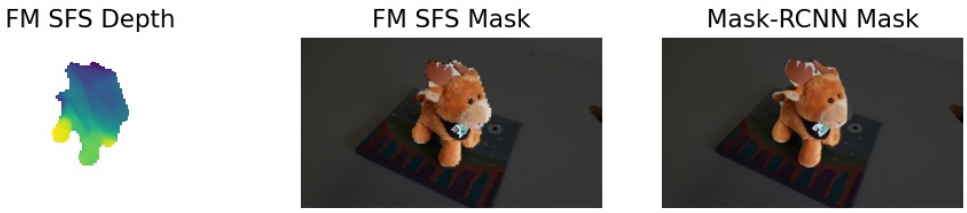
Here we describe additional details about the experiment shown in Fig. 9 of the main chapter. Concerning the differentiable renderer: the method, settings and hyperparameters are identical to those in Fig. 7 and 8 and Section 5.3. We simply run the method on different input.

We collected a 14 second video with a Samsung S9 cell phone at 1280 x 720 resolution at 30 Hz. The video contains motion blur, auto-exposure, and clearly visible video compression artifacts, making it unsuitable for some reconstruction methods. We sub-sampled the video down to 6Hz and ran Mask RCNN [He+17] from Detectron2 [Wu+19] with the pre-trained weights `COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml` to detect objects. In our case, the object was detected as part of the teddy bear class, with about 55 viable frames. We ran COLMAP [Sch+16] to obtain camera poses for those frames, where COLMAP successfully returned 36 frames with valid camera poses. We ran our SFS pipeline at 160 x 90 resolution to obtain the results shown. Visual examples from this pipeline are shown in Fig. 2.10. All the methods used their default settings; there was no parameter tuning involved.

2.10.1 Video Result Analysis

The trajectory shown here only covers about half of the object from a roughly constant elevation. Complicating the reconstruction is that the camera poses are imperfect due to estimation and unmodeled camera distortion. Much more significant is that the Mask RCNN silhouettes used for reconstruction are often extremely under or over segmented.

Despite these issues in the "ground truth" used for optimization, the low degree of freedom of Fuzzy Metaballs allows the model to reasonably recover from the massive artifacts. While the results in this chapter typically show the default 50% threshold, to recover some areas, we have to lower our α threshold to 10%.



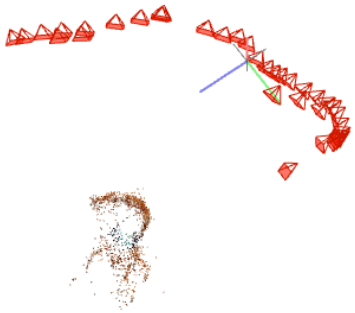
(a) **Depth and silhouette** from a shape-from-silhouette reconstruction.

(b) **Recovering from undersegmentation in the ground truth masks.** While a 50% threshold does a good job recovering the head, better recovery can be shown with a 10% threshold, also recovering the leg.

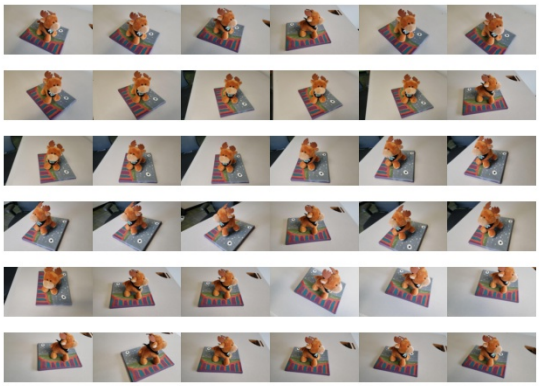
(c) **Recovering from oversegmentation in ground truth masks.** Even the $\alpha = 10\%$ threshold only leads to minor over-segmentation in the mask, suggesting a setting that be appropriate in general.

Figure 2.9: **Shape from silhouette reconstruction on natural images** from a handheld cell phone video, using COLMAP [Sch+16] and Mask RCNN [He+17] for automatic camera poses and silhouettes. The low degree of freedom leads to natural regularization and recovery from errors in ground truth.

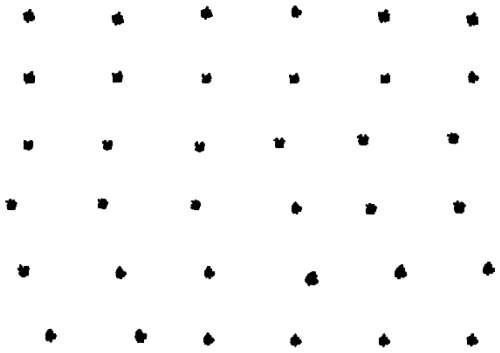
(a) Mask RCNN output for valid frame



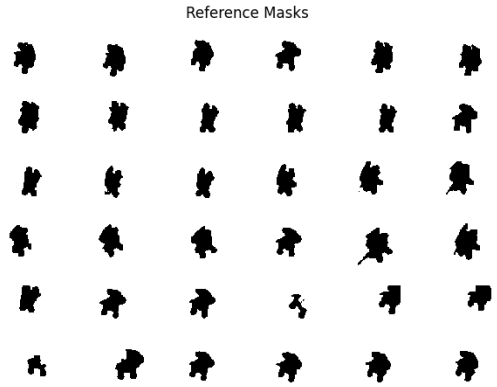
(b) COLMAP estimate of camera poses



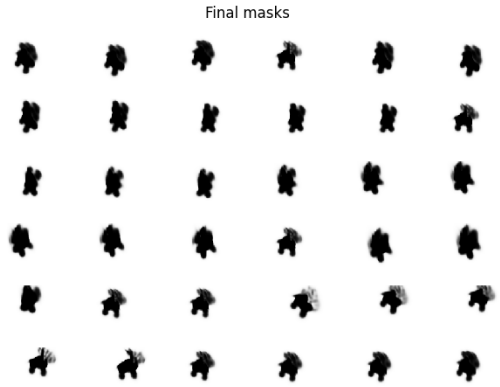
(c) All 36 frames used for SFS



(d) SFS initialization



(e) Mask RCNN Silhouettes



(f) SFS Mask Results

Figure 2.10: **Video-based SFS reconstruction**

2.11 Hyper-parameters

Our proposed method has 5 hyper-parameters described in this chapter. Briefly, β_1 prioritizes close hits, β_2 prioritizes hits closer to the camera, β_3 prioritizes hits in front of the camera, and β_4 and β_5 serve as a scale and offset to generate alpha masks. Since our system is fully algebraic, it is possible to perform gradient descent into these hyper-parameters (and the functional form of JAX naturally returns their gradients), but this was not done.

Instead, we optimized them for depth and alpha mask accuracy over a small simulated dataset of the Stanford bunny using standard black-box optimization techniques [Han16; HAB19; LH16] before running most of our experiments. We found that the ray-based renderer led to similar optimal hyperparameters across multiple tested resolutions, across a wide range of mixture components, and across our linear, quadratic and cubic methods of intersection computation.

2.12 Exclusion of `gear` model

The `gear` model was selected because of its interesting geometry from Thingi10k [ZJ16]. However, for pose estimation, we exclude its results from the overall average due to symmetry. Our poses are generated with rotations of uniform axis and angle uniformly between -45 and 45 degrees (uniform-axis random spin [Sta14]). The gear model however has 15 teeth and a rotational symmetry of 24 degrees when viewed from one side, as seen in Fig. 2.11. This can sometimes produce pose errors with no real geometric error.

The model itself is not symmetric, with 15 gears and a back face with 180 degree symmetry. But with a single view, our testing conditions can generate poses which are geometrically correct but produce pose errors. The other model with symmetry, `eiffel`, only has 90 degree symmetry and our testing conditions place all random poses in the same local minima.

We don't use the `yoga` or `plane` models for pose estimation as we only latter added them for the reconstruction experiments. Both models originate from prior differentiable rendering uses in reconstruction [Liu+19; Yif+19].

2.13 Pose Estimation Details

We include noise-free results the same seed as the noisy results earlier in this chapter. Summary plots are shown in Fig. 2.12 and Fig. 2.5. In the noise-free case, we find that Point-to-Point ICP works better. With noise, Point-to-Plane ICP methods perform better.

2.13.1 Noise Free

As described above, when ICP methods perform well, they perform extremely well, an order of magnitude better than the differentiable renderers (see the log-scale plot), to fractions of a degree since they have high resolution samples. However, sometimes ICP finds poor local minima and on average our method performs better, even when ICP has a dense point cloud. Despite having a better mean, Fuzzy Metaballs (FM) have a median error that is 8 times higher and a 25th percentile error that is 10 times higher. The increase in robustness from FM is demonstrated in lower 75th percentile errors.

2.13.2 Noisy Depth Images

With synthetic noise, both differentiable renderer methods are barely affected, while the ICP results see a large degradation in peak performance. Under this experimental condition, Fuzzy Metaballs have the lowest mean, median, 25th and 75th percentile errors (typically by a factor of 2 compared to ICP).

Interestingly, some of the worst case performance of the ICP methods disappears (lower q3 measurements) when noise is added. We hypothesize that this occurs due to a form of symmetry breaking that helps avoid singularities and bad correspondences. Fuzzy Metaballs, being a low fidelity model, experience nearly no degradation in performance when noise is added to depth images.

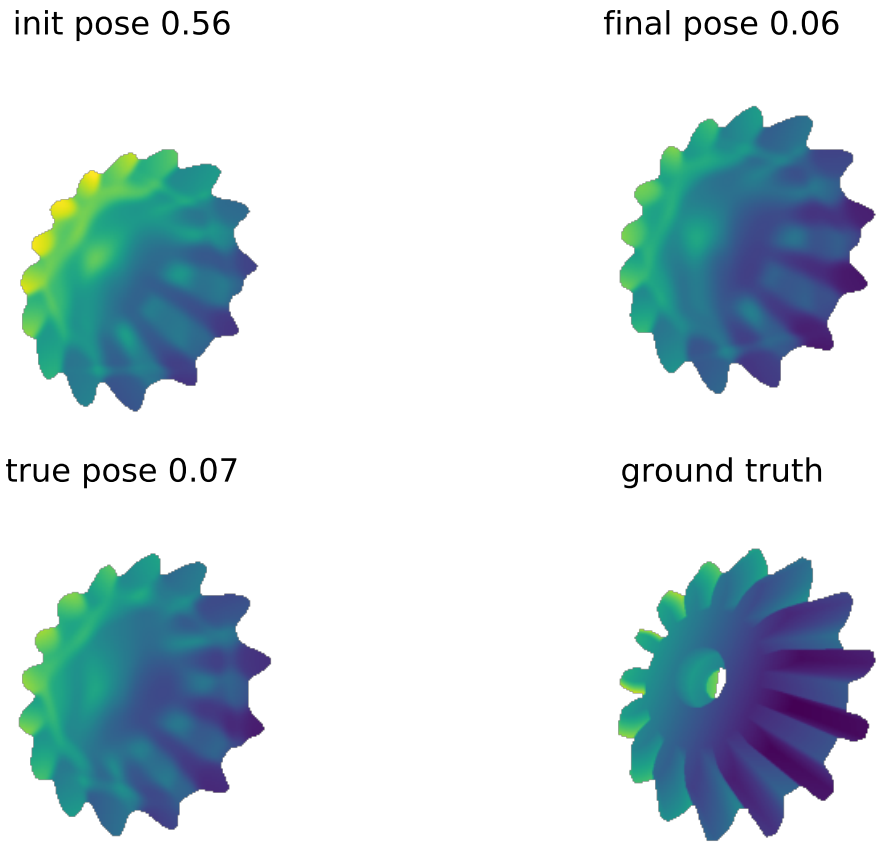


Figure 2.11: **Gear Results with Fuzzy Metaballs** Final pose describes the final pose after gradient-based pose optimization, while true pose is rendered view from the ground truth pose. Ground truth is the Blender-generated depth map of the full-fidelity model. The final pose shown here has a rotational error of 23.9 degrees. However, the gear has 15 teeth and hence a 24.0 degree symmetry.

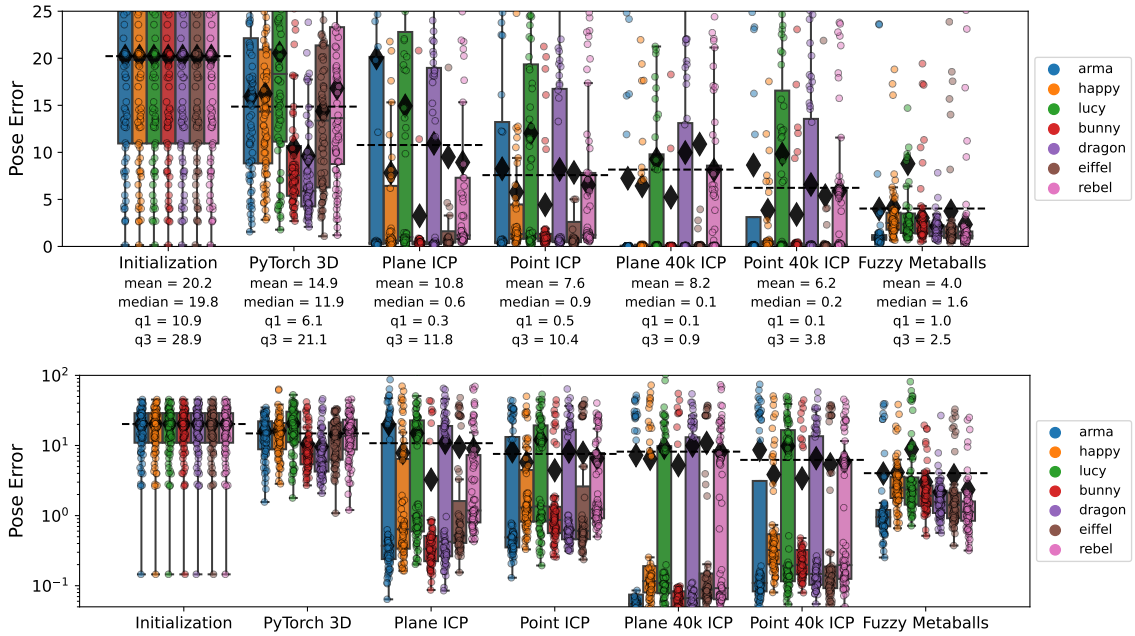


Figure 2.12: **Noise Free Pose Estimation** Linear scale plot above and log-scale below. Dashed lines are averages for the method, while the black diamonds show the average for that method and model. Statistics for each method are listed. `gear` model is excluded from statistics.

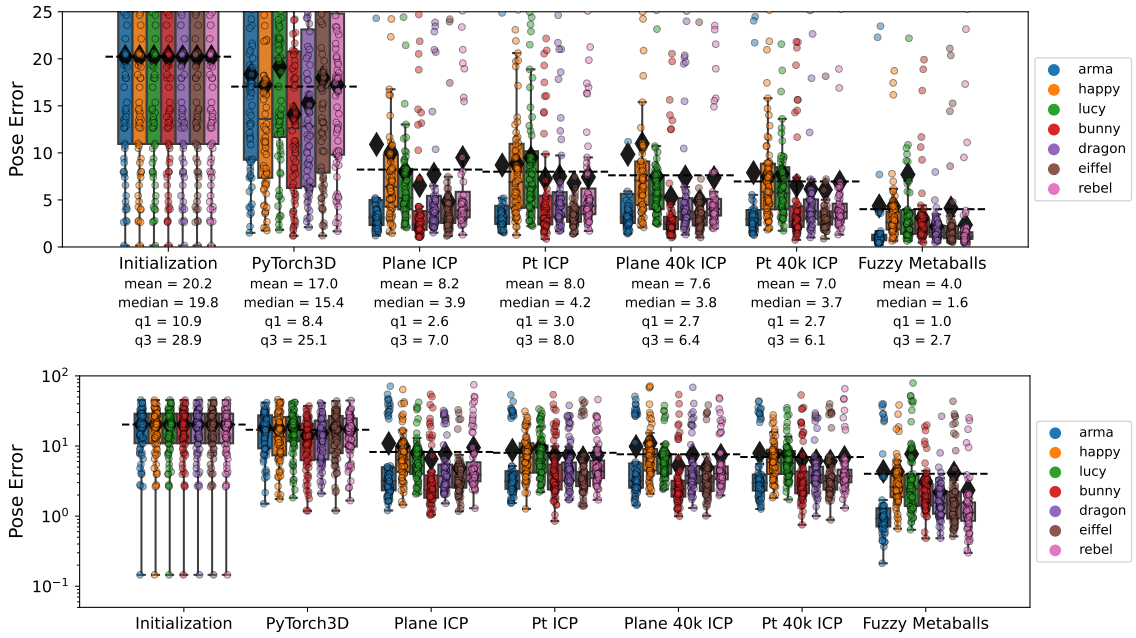


Figure 2.13: **Noisy Pose Estimation** Identical visualization to Fig. 2.12. Linear scale figure is identical to that shown earlier.

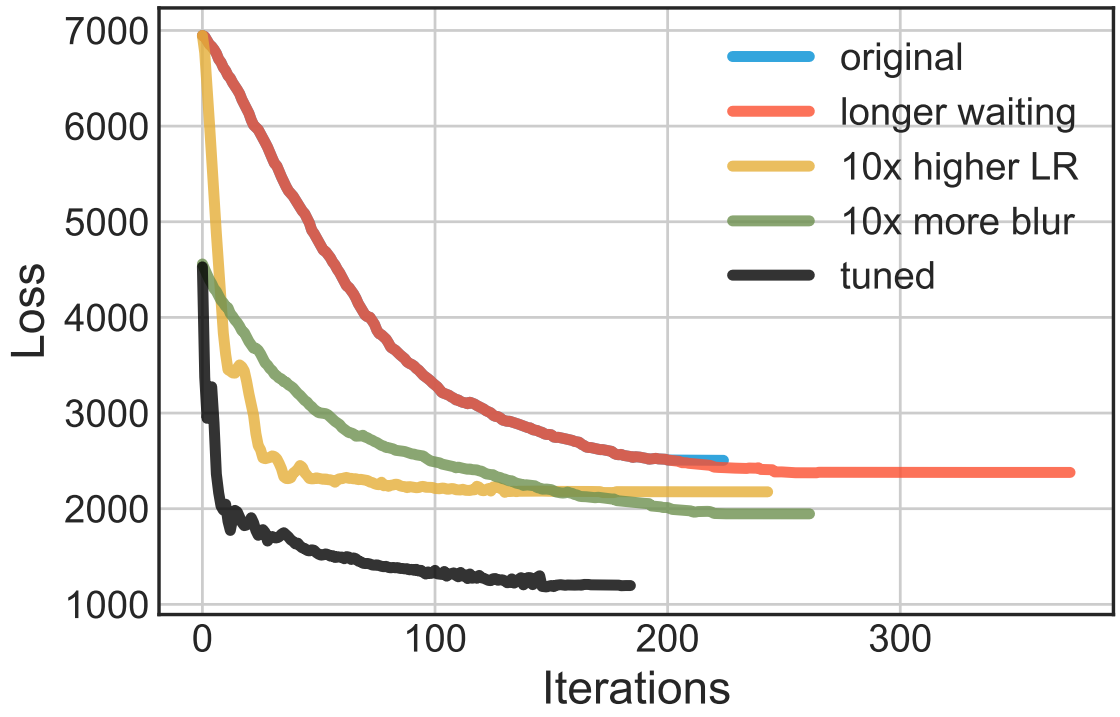


Figure 2.14: **PyTorch3D baseline** convergence curves using different hyperparameters for pose estimation. All curves produce roughly equivalent pose errors, significantly worse than FM or ICP.

2.14 SoftRasterizer performance

One might wonder about why our baseline of PyTorch3D, implementing SoftRas [Liu+19], performs so poorly in the pose estimation experiments. Prior work on differentiable rendering [Liu+19; LZ21; Yif+19] demonstrates their pose optimization experiments with mostly single, visual examples. These often use color images, and provide no baselines results from standard methods. Quantitative results in SoftRas [Liu+19] examined solving for rotation uncertainty in images of a colored cube and their resulting rotation errors averaged over 60 degrees. It is perhaps not surprising that our pose estimation experiments, featuring a family of models, simultaneous rotation and translation, while optimizing only depth and silhouettes, might challenging these methods.

We used the hyperparameters from the current PyTorch 3D [Rav+20] *Camera*

position optimization sample. We tuned learning rates to behave well with our depth + silhouette loss function, and followed an automatic learning rate schedule [Kin18]. As can be seen in Fig. 2.15, the pose optimization performs reasonably well in reducing image errors. However, the optimized pose still demonstrates visual errors compared to the ground truth pose. Even worse, the optimization perturbs the pose in such a way that the pose error at the end of optimization (16 degrees and 15%) is worse than the pose errors at the perturbed initialization (12 degrees and 8%), despite the significant reduction in loss.

To check if the hyper-parameters from the PyTorch3D sample was a poor fit, we searched for settings which produced good pose estimation for a single frame. We used CMA-ES [Han16; HAB19], a fairly common black box method [LH16]. This type of task-specific hyper-parameter optimization was **never performed** for our Fuzzy Metaballs experiments. We only performed these experiments on an existing baseline to examine how good it might perform in the best case. Convergence curves can be seen in Fig. 2.14.

All our manual tweaking of PyTorch3D hyper-parameters produced comparable configurations (17-18 degrees of rotation, 15-16 percent translation). The automated optimization found a setting which produced 16 degrees of rotation error and 9 percent of translation error, still worse than the perturbed initialization. However, these settings used a very high learning rate that proved unstable with other frames. Lowering to learning rate resulted in settings with a negligible improvement (2%) to our initial settings. These tests suggest our initial hyper-parameter choices were a reasonably good setting for the baseline method.

Further parameter search with a constraint on learning rate failed to find parameters significantly improved from the defaults. Optimization was over 8 parameters: σ, γ, blurring radius for both depth and silhouette, faces per pixel, learning rate, and multipliers for depth and silhouette loss.

The results of the PyTorch3D pose optimization, when fed into the Fuzzy Metaballs renderer suggest these implementations are coherent– the FM optimization run on the same original pose perturbation and model produces a 0.5 degree error and 1.5% pose translation error, while even noisy point-to-point ICP gets a 2.6 degree and 4.3% translation error.

Both the Fuzzy Metaballs and PyTorch3D optimization use an axis-angle, 3

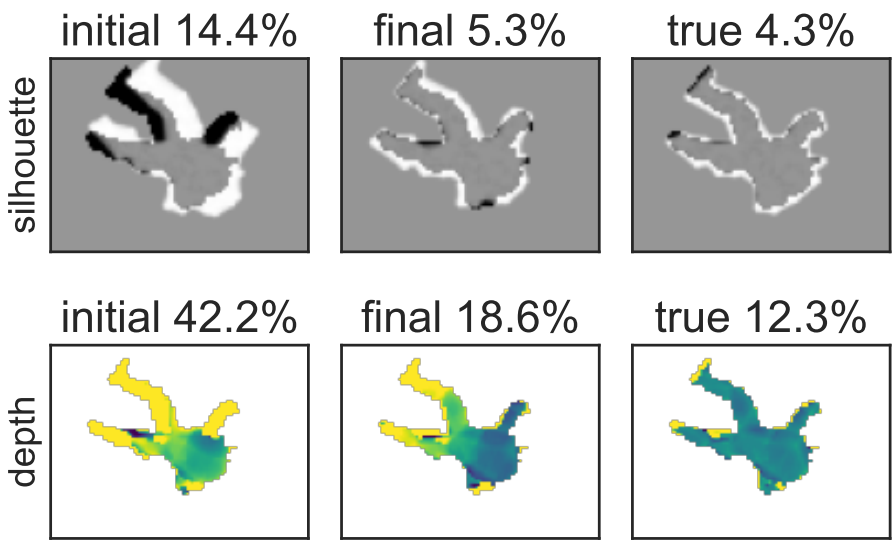


Figure 2.15: **PyTorch3D baseline** Visualization of errors seen with pose optimization. Initial is an example pose perturbation of the `arma` model. Final is the result after pose optimization, and true is the result of the ground truth pose. Silhouette error is in percent of pixels that are wrong, while depth error is in average relative depth error. Optimization leads to a reasonable decrease in both.

parameter rotation estimation. These is some evidence suggesting PyTorch Autograd for $SO(3)$ might be unstable at times in its native form [TD21b]. Lastly, we suspect that the same scale invariance issues we address in Section 2.7.1 may exist for the PyTorch3D baselines. SFS experiments were performed on objects of roughly unit scale to mitigate this potential issue.

2.14.1 Pulsar performance

Our attempts to test a recent differentiable renderer, *Pulsar*, found it performed very poorly. Not only are there software bugs with the latest PyTorch3D at the time of writing (`0.6.1`) where the code clobbers camera data-structures and requires re-creating them with every call to the render function, but the pose estimation results were very poor.

We used the same settings as the Point Cloud Differentiable Renderer baseline we tested, which provided fair results and produced visually similar outputs. Compared to our base learning rate, reducing it by a factor of two led to flat loss. Increasing it

by a factor of two led to divergence and NaNs.

2.15 Exporting Fuzzy Metaballs

We experiment with exporting fuzzy metaballs as a mesh by running marching cubes [LC87]. To find an ideal isosurface level, we run optimization to ensure that the centroids of the voxels match the silhouettes over a sample set of views. This leads to results like those in Fig. 2.16.

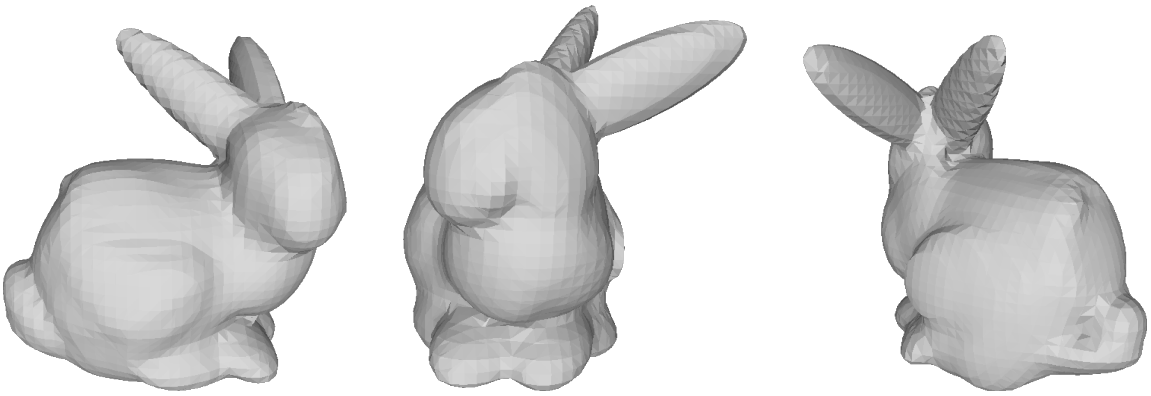
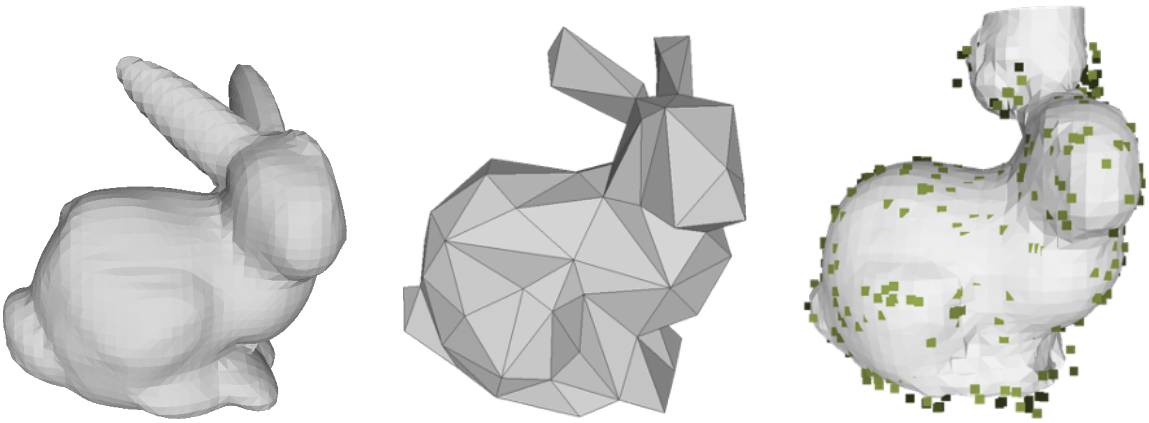


Figure 2.16: Mesh extracted from a 40 mixture fuzzy metaball using marching cubes

2.16 Fuzzy Metaballs as Surface or Volume GMMs

To understand what classical formulation best matches Fuzzy Metaballs, we try optimizing models with different initializations. We start with both sphere and EM-fit GMM initializations, with surface and volume versions of both. Quantitative results averaged across all 10 models are shown in Fig. 2.19. Qualitative results for the `Yoga` model are shown in Fig. 2.23.

At low mixture numbers, Fuzzy Metaballs perform more like a volume GMM, while at high mixture numbers, surface GMMs work better. Often using a GMM as a FM model will produce reasonable results. We use constant hyperparameters from our 40 mixture tuning, and perhaps the out-of-the-box vGMM rendering could



(a) 40 component Fuzzy Metaballs (400 params)

(b) 170 face, 85 vertex Mesh (810 params)

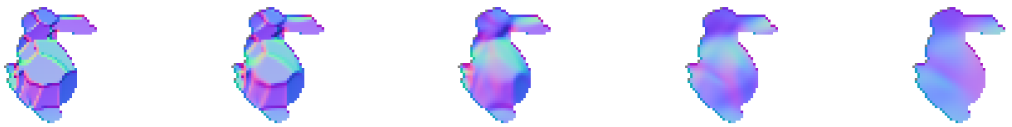
(c) 430 Points reconstructed [KBH06] (1290 params)

Figure 2.17: Equivalent representations visualized, per the experiments in the chapter.

improve by adding proper scaling with component number. Finally, all initializations respond very well to optimization, and optimized sphere-initialized models always outperform the models fit with solely with EM.

The Fuzzy Metaballs improve with more components across our entire range of testing. This suggests the asymptotic behavior seen in the **Comparing Representations** section is due to experimental factors of those experiments, and not the representation itself. This is somewhat expected as those experiments use the mesh representation as ground truth and all other formats are sampled.

Lastly, we can see that the fitting process produces no over-fitting as novel and training frames have identical behavior in Fig. 2.22.



(a) Visualizing normal maps while sweeping β_1 and β_2 demonstrates smoothing.

(b) Sweeping β_4 and β_5 controls the sharpness and extent of the alpha masks.

Figure 2.18: Hyperparameter visualization

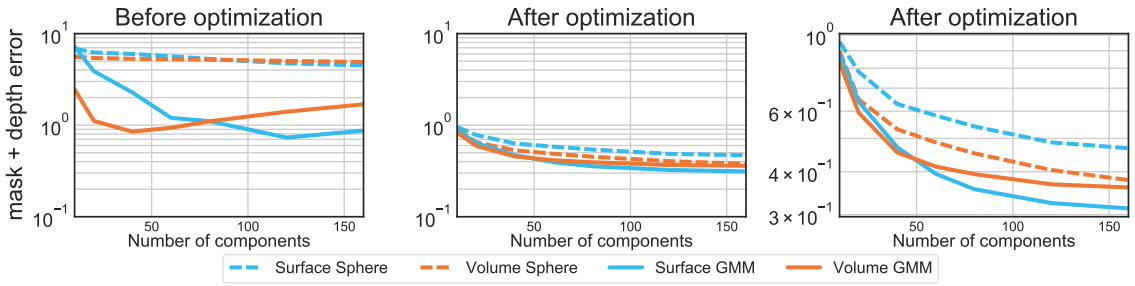


Figure 2.19: Optimizing Fuzzy Metaballs from different initializations.

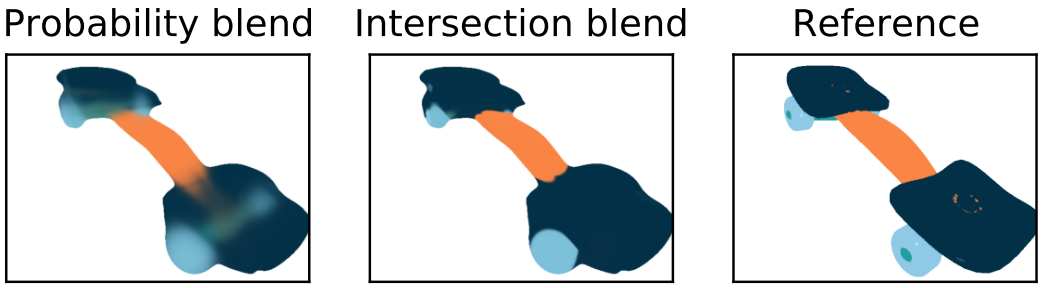


Figure 2.20: Rendering Fuzzy Metaball color images of a snakeboard [KCD09] with two forms of blending: one behaves more like a volume where the wheels of the object can be seen, while the other behaves more like a surface with proper occlusion. Shown is a 40 component vGMM with a single color per component. Cartoon-like appearance is from exclusively using ambient lighting.

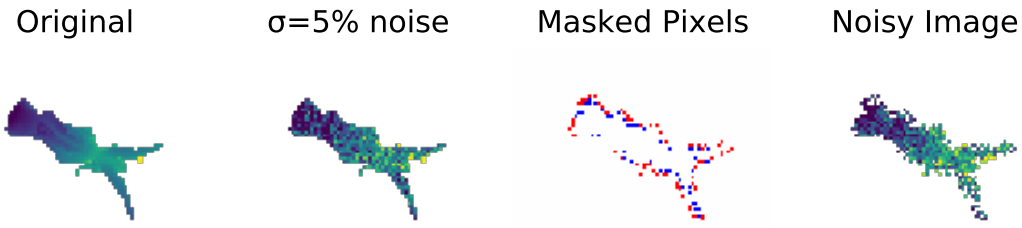


Figure 2.21: **Synthetic noise generation**. Gaussian noise is combined with perturbed silhouettes (red pixels are added, blue are removed).

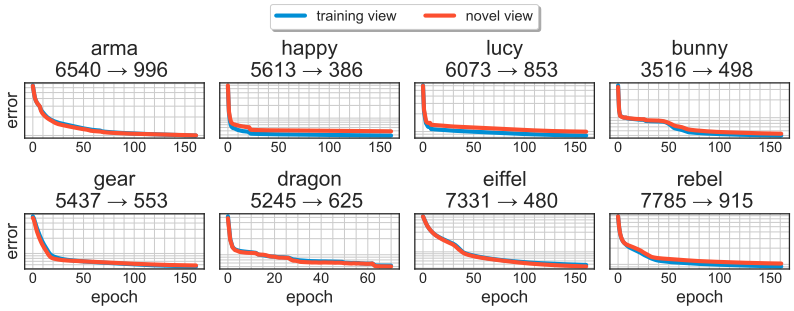
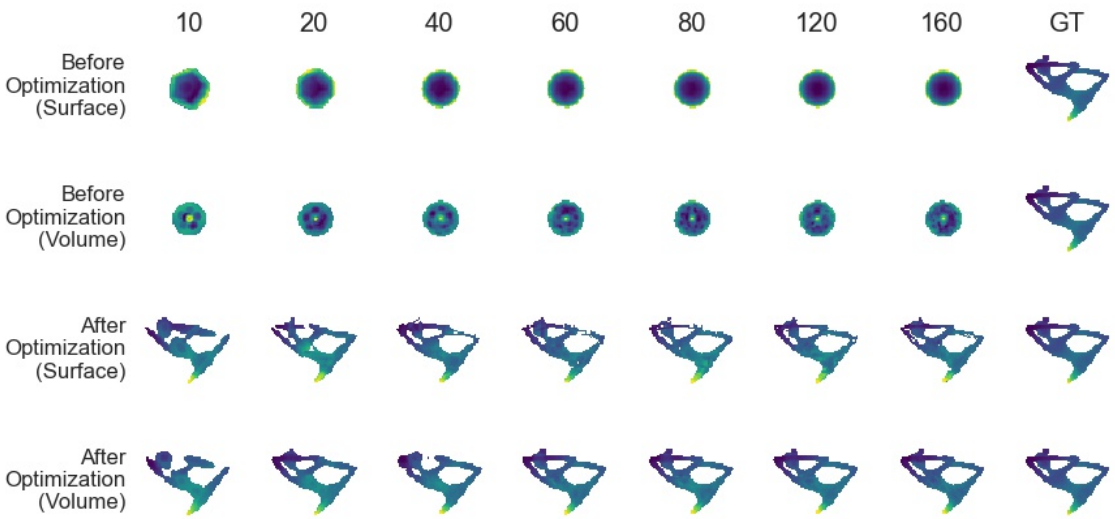
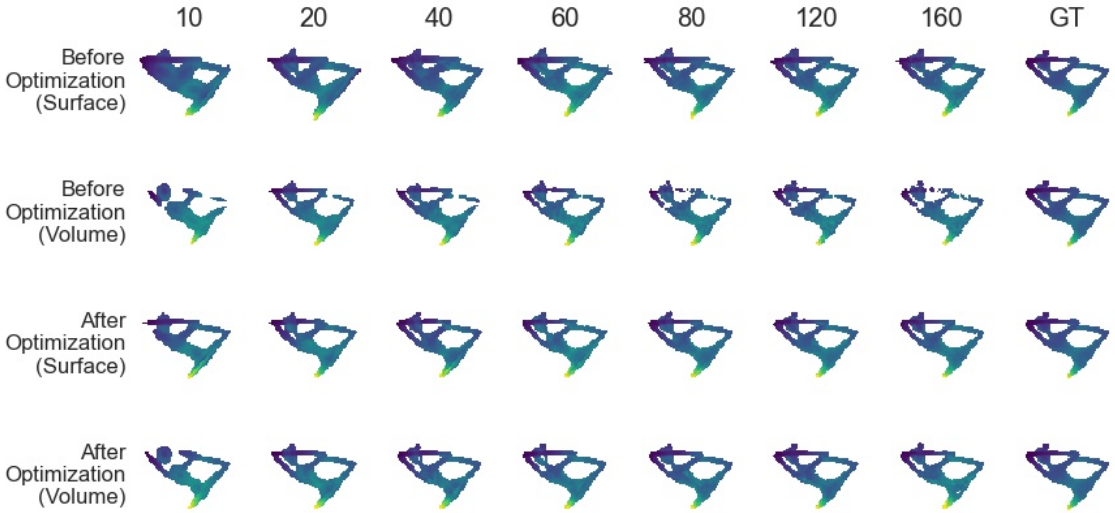


Figure 2.22: Optimizing Fuzzy Metaballs from a sphere to a shape. Losses are given for training frames and novel viewpoints, showing no significant difference.



(a) Sphere Initialization



(b) GMM Initializations

Figure 2.23: Visual examples of Fuzzy Metaballs at different component numbers, for different initializations, before and after optimization. All images are 60 by 80 pixels and show depth with color coding. Here, unlike the rest of the chapter, colors are scaled for maximum contrast, not consistency between images. GT is the ground truth depth map from the mesh rendered by Blender.

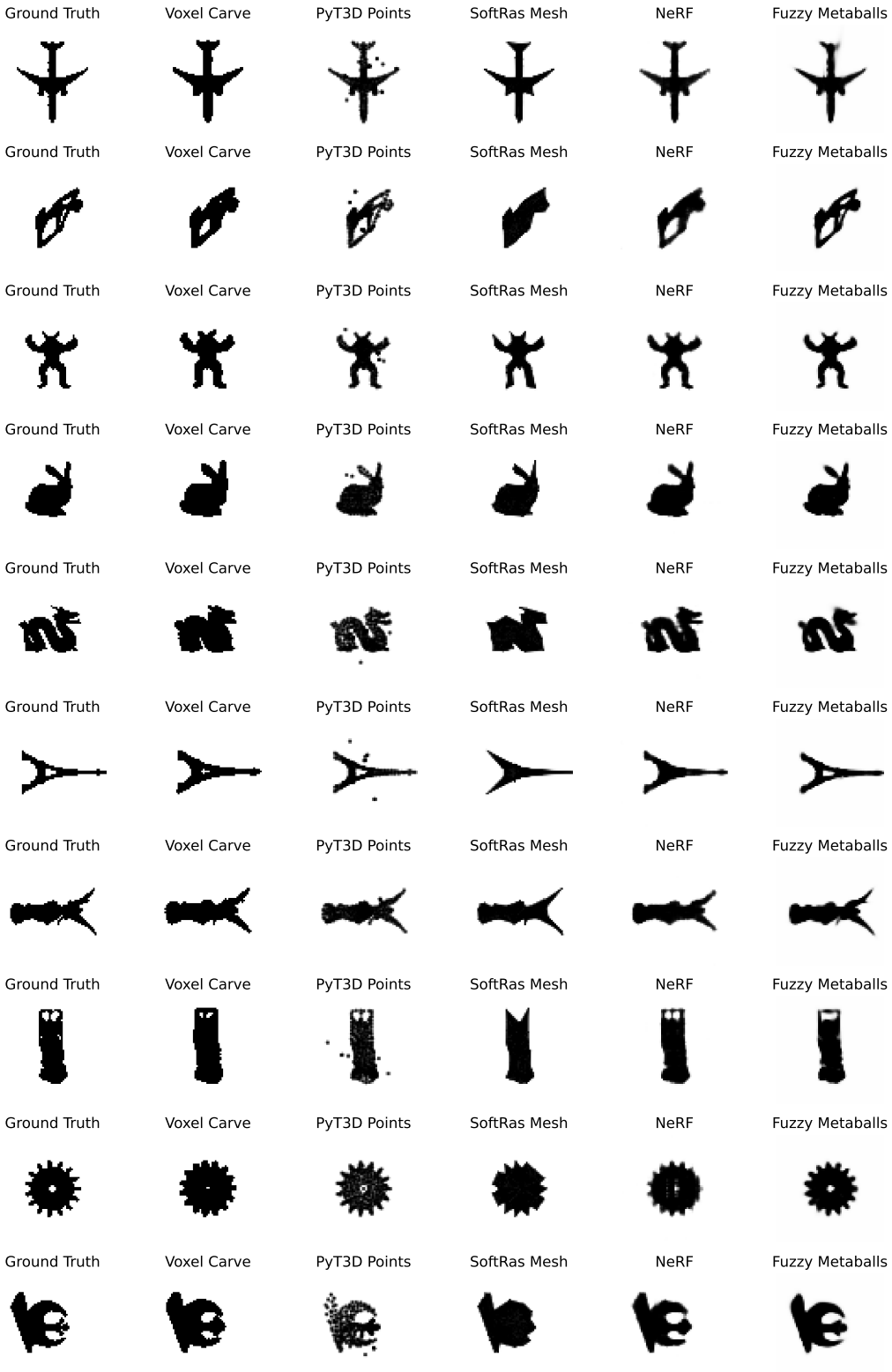


Figure 2.24: **Shape from Silhouette Results**. The mesh-based representation cannot change genus from a deformed sphere into the eiffel tower. The point cloud method leaves spurious points. The classic Voxel Carving method is not that precise with 384^3 volume but only 32 views of low resolution 64 x 64 images.

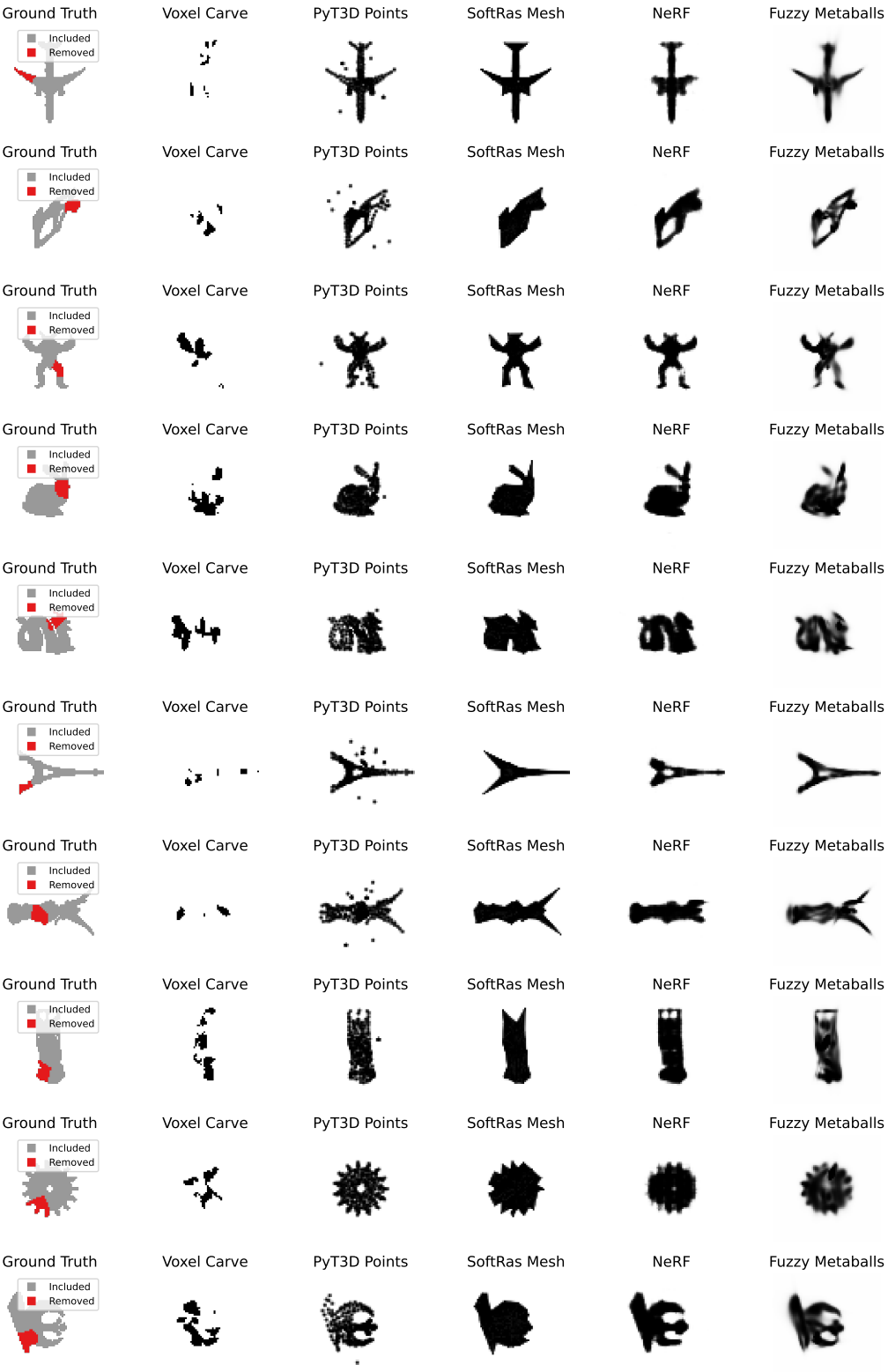


Figure 2.25: **Shape from Silhouette Noisy Results** where 16 of the 32 input views had one eight of the silhouette under-segmented.

Chapter 3

Flexible Uses of 3D Gaussians

In this chapter, we simplify and extend the techniques of Chapter 2, and our presentation is largely similar to a conference manuscript of this material [KH23b].

Here, we produce useful extensions for 3D Gaussians used in shape reconstruction (removing unneeded parameters, introducing optical flow loss, and comparing sorted and sort-free rendering approaches). Additionally, we show how to export colored meshes and robustly re-parameterize Gaussians to add detail and remove low impact Gaussians.

While our earlier chapter introduced 3D Gaussians for differentiable rendering to the literature, we also discuss connections with follow-up work that occurred since our initial publication. A year after our publication, 3D Gaussian Splatting [Ker+23] built a faster rasterizer for 3D Gaussians, enabling larger scale experiments and showing NeRF [Mil+20a] levels of fidelity. We show that FM and 3DGS are capable of rendering the same underlying 3D Gaussians and hence are largely the same representation, despite some differences in rendering approach.

3.1 Introduction

As computer vision systems are more widely deployed in society, either on robots or via mixed reality headsets, users will desire that reconstructions of their many regular everyday objects. While classic techniques from multiview scene reconstruction could be used [Sch+16], modern approaches strive for more photorealistic scene

generation, such as those created by Neural Radiance Fields (NeRF) [Mil+20a] and the large array of follow-up work [Tan+23]. Of note, these NeRF approaches could be seen as differentiable renderers [Liu+19; Yif+19], where an underlying scene representation is optimized for view synthesis. However, NeRF methods are demanding in their computational requirements, even with speed-up methods such as Instant-NGP [Mül+22].

Two recent papers explored similar approaches to fast NeRF alternatives – Fuzzy Metaballs [KH22] (Chapter 2) in 2022 and 3D Gaussian Splatting in 2023 [Ker+23]. Both approaches represent geometry using a set of classic primitives, specifically 3D Gaussians [Hec86; Pea94]. The former built a differentiable raytracer for 3D Gaussians, connected it to the metaball literature [Bli82], developed a sort-free rendering function, focused on fast CPU runtimes, used dozens of Gaussians, and performed quantitative experiments for object reconstruction and pose estimation. A year layer, the latter designed a fast rasterizer with a custom CUDA kernel for splatting [Man06; NM01], used millions of Gaussians, and focused their system on reconstructing entire scenes to approach NeRF levels of fidelity. We demonstrate connections between and extensions to both methods.

This chapter is focused on extending the Fuzzy Metaballs renderer to make it simpler, more robust, and to add additional features. We show that these recent approaches are interoperable and render the same underlying representation. Since most well-established rendering techniques are built on triangle meshes, we demonstrate a reliable way to transform 3D Gaussian representations into meshes.

We summarize our contributions as follows:

- Develop a simplified version of Fuzzy Metaballs [KH22] for shape reconstruction from Gaussians (Section 3.3.2).
- Show how Fuzzy Metaballs [KH22] can be rendered without hyperparameters (Section 3.3.3).
- Show how to get per-pixel, differentiable optical flow and its benefits in reconstruction (Section 3.5).
- Demonstrate how to export meshes from shapes defined by 3D Gaussians (Section 3.6).
- Show that existing 3D Gaussian rendering methods [Ker+23; KH22] are inter-

(a) **Fuzzy Metaballs** [KH22] Chapter 2 raytraced dozens of Gaussians, used random initialization, and estimated geometry and pose for objects.

(b) **3D Gaussian Splatting** [Ker+23] splatted millions of Gaussians, used SfM [Sch+16] initialization, and synthesized novel views for scenes.

Figure 3.1: **Recent approaches in differentiable rendering use 3D Gaussians as an underlying representation.**. These approaches enable fast reconstruction of 3D objects and scenes. This chapter demonstrates how both approaches use the same underlying representation and how to enhance and export such reconstructions.

operable (Section 3.7).

- Develop a loss-based approach to reparameterizing Gaussians by splitting components (Section 3.8).

These techniques enable more powerful, flexible uses of these 3D Gaussian shape representations for applications involving shape reconstruction. Of note, these approaches require no pretraining and can be optimized directly on the scene of interest without dataset bias, making them applicable to robotics applications where robots may be interacting with novel objects, and are limited by onboard compute.

3.2 Related Work

A comprehensive overview of related work can be found in prior papers using 3D Gaussians, including Fuzzy Metaballs [KH22], 3D Gaussian Splatting [Ker+23] and VoGE [Wan+23].

Some early methods of building models from partial observations used generalized cylinders [Agi72]. More commonly, methods build on top of triangle meshes, point clouds and surfels [Pfi+00]. Differentiable renders have been built for these representations, initially for meshes [KUH17; Liu+19; LB14]. These include custom backends that allow for fast GPU-based results [Lai+20], and high-quality results [Nim+19]. Other works focus on point clouds [ID18; Yif+19]. Pulsar [LZ21] uses spheres as its representation, which are equivalent to isotropic 3D Gaussians. Primitives-based rendering methods have benefits for both for composition [Zha+23] and tracking [Lui+23].

The earliest work using 3D Gaussians in rendering came from Blinn [Bli82], originally called atoms, blobs or metaballs, and were the birth of implicit surfaces. Several methods these techniques [Gou+10; Hor19; Mur91; SI12; Wes90; WMW86; WT90]. Some renderers used rays while others used splatting [ALD06], and some recent differentiable renders build screen space Gaussians [MWK22]. Others use Gaussians as a primary representation in computer vision [EKK18; Eck+15; Eck+16; Gen+20; GMT23; Her+20; Mag09] or render them via projection, search, or other techniques [Hua+20; OTM19; SM20].

Gaussians can be seen as a fundamental building block that only uses the 1st

(μ) and 2nd (Σ) order moments [JUD00]. Point clouds only use μ. Oriented point clouds add a covariance eigenvector [EKK18], and Gaussian Mixtures [Eck+16] use all the information. Connections in this space include error metrics [GH97; Mah36] and physics simulations [Mir96].

There is work on connecting NeRF-style differentiable renderers to mesh representations. MobileNeRF [Che+23] used the rasterization pipeline of commodity hardware to perform rendering of NeRF-like objects. VMesh [Guo+23] constructed a hybrid representation of volume and mesh. NeRFMeshing [Rak+23] learned a Signed Surface Approximation Network to distill NeRF representations into meshes.

3.3 Ray-Shape Intersections

Existing methods for intersecting 3D Gaussians and rays typically take two forms. Methods taking inspiration from NeRF family methods [Ker+23; Wan+23] typically sort all intersections and use closer Gaussians to attenuate the contributions of further Gaussians. Fuzzy Metaballs [KH22] introduced a heuristic technique for blending intersections which did does not require sorting. Section 3.3.1 summarizes this technique, Section 3.3.2 presents a simplification, and Section 3.3.3 proposes a variation without hyperparameters. All three correctly render the same shapes, so objects optimized by one can be reproduced with the other.

3.3.1 Weighted Blending

In Fuzzy Metaballs [KH22], intersections between each ray ($\vec{v}$) and each Gaussian are computed separately and then combined with a weighted average. Each Gaussian is parameterized as mean ($\mu \in \mathbb{R}^3$), inverse root precision ($\Sigma^{-\frac{1}{2}}$) and a weight ($\lambda_i \geq 0$) where the $\sum_i \lambda_i = 1$.

Multivariate Gaussians are defined as

$$P(\vec{x}) \propto |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\vec{x}-\mu)^T\Sigma^{-1}(\vec{x}-\mu)\right), \tag{3.1}$$

and the unnormalized log distance for each Gaussian is

$$d(\vec{v}t) = -\frac{1}{2}\left[(vt-\mu)^T\Sigma^{-1}(vt-\mu)\right] + \log(\lambda_i), \tag{3.2}$$

which we will refer to as d_i when referring to evaluating the i-th Gaussian for a ray $\vec{v}$ (here we use the point of maximum likelihood, $t_i = \frac{\mu_i^T\Sigma_i^{-1}v}{v^T\Sigma_i^{-1}v}$, the linear approach [KH22]).

For each Gaussian, its intersection (t_i) and distance (d_i) are used to obtain weights (w_i). The final intersection is obtained as t_f, similar to OIT [End+10; MB13]:

$$t_f = \frac{1}{\sum_i w_i}\sum_i w_i t_i. \tag{3.3}$$

Per-ray estimation of other properties can continue to use Equation (3.3). For example, t_i (distance from camera) can be replaced with $\vec{n_i}$ (normal) or $\vec{c_i}$ (color), or even learned features $\vec{z_i}$ (e.g. DINO [Car+21; Oqu+23] or CLIP [Rad+21; Ilh+21]), and the same blending functions can be reused. For more details about normal computation, see Section 3.6. It is also helpful to think of the unnormalized Gaussian density as:

$$\delta_i = \exp(d_i), \tag{3.4}$$

The original Fuzzy Metaballs approach uses 5 hyperparameters ($\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$) to compute the weights and the transparency. There is also a shape scale (η) to account for shapes of different scales and return identical results. The following are the weight and α (quality/opacity) functions:

$$w_i = \exp\left(\beta_1 d_i \sigma\left(\frac{\beta_3}{\eta}t_i\right) - \frac{\beta_2}{\eta}t_i\right), \tag{3.5}$$

$$\alpha = \sigma\left(\beta_4\sum_i \delta_i + \beta_5\right). \tag{3.6}$$

3.3.2 Two Parameter Model

In shape reconstruction from video, 3 of these hyper-parameters are not necessary and we develop a simplified two parameter model that works in reconstruction settings.

First, β_3 was used to give some minor contribution to intersections behind the camera. While this does make the renderer more differentiable, it is not used in practice, so the σ function can be removed, leaving us with

$$w_i = \exp\left(\beta_1 d_i - \frac{\beta_2}{\eta} t_i\right). \tag{3.7}$$

Since the earlier chapter Chapter 2 focused on rendering full, proper Gaussian Mixture Models (where $\sum_i \lambda_i = 1$), it required a normalizing factor β_4 to account for how much opaque a GMM should be. In the case of reconstruction, there is no need for strict GMMs, so learning β_4 can be the responsibilities of unnormalized λ_i. Additionally, the GMM focus suggested a smooth step function (sigmoid) for computing α, but shape reconstruction can simplify this to a decaying exponential, letting us drop the intercept term β_5. This allows us to have a simple, perhaps familiar [Mil+20a; VJK21], expression for α:

$$\alpha = 1 - \exp\left(-\sum_i \delta_i\right). \tag{3.8}$$

We use this simplified two parameter model in most of our experiments. While we noticed some small differences due to losing the global normalization condition, they were minor and mostly in the space of pose estimation, which is not the focus of our experiments here for shape estimation.

3.3.3 Zero Parameter Model

With the simplified design given above, and the NeRF-style approaches in other 3D Gaussian papers [Ker+23; Wan+23], we also investigate an alpha compositing variant of Fuzzy Metaball rendering that requires sorting all intersections and then computing transmission. The transmissions can be seen as weights and Equation (3.3) can be used to compute depth estimates (as well as normal and color estimates):

$$w_i = T_i \left(1 - \exp(-\delta_i)\right) \tag{3.9}$$

where T_i is the accumulated transmissions of earlier intersections

$$T_i = \exp\left(-\sum_j \delta_j 1[t_j < t_i]\right) \tag{3.10}$$

These equations create a version of differentiable rendering with metaballs that is hyperparameter-free, doesn't require splatting, and enables the same usage of weights for depth, normal and color computation as prior work [KH22]. As our experiments in Table 3.2 and Figure 3.7 show, this approach is slower but performs nearly identically.

Figure 3.2: **Shape reconstruction** using 40 3D Gaussians and converging in under one minute, with color. (See Section 3.4 for details). All objects are reconstructed from videos in the CO3D [Rei+21] dataset.

3.4 Shape Reconstruction

The input to the system is a video and a single masked frame. COLMAP [Sch+16] produces poses, XMem [CS22] propagates masks, Unimatch [Xu+23] produces flow, and 3D Gaussians optimized to fit the data. The shape converges quickly since we use a ray-based differentiable renderer and are able to sample minibatches that includes pixels across all frames.

Our differentiable rendering code is based on Fuzzy Metaballs [KH22], which is built in JAX [Bra+18] and allows for reconstructions on both the CPU and GPU. With an Nvidia GTX 1080, we can do memory for image sequences of roughly

250×130, while our CPU experiments are typically closer to 125×65. Both sets of experiments typically converge in less than a minute on commodity hardware. We use 40 Gaussians as in prior work, for ease of comparison.

Differentiable renders provide flexibility for many loss functions in shape reconstruction. For our experiments, we combine of cross-entropy loss (L_M) for objects segmentation masks, and L_1 losses for color and flow (Section 3.5), weighted by the object mask. Estimated alpha is clipped to $[10^{-6}, 1 - 10^{-6}]$. For α, color (c), and optical flow (f) we use the following loss function:

$$L_M = \alpha \cdot \log(\hat{\alpha}) + (1 - \alpha) \cdot \log(1 - \hat{\alpha}) \tag{3.11}$$

$$L_C = \alpha \cdot ||c - \hat{c}||_1 \tag{3.12}$$

$$L_F = \alpha \cdot ||f - \hat{f}||_1 \tag{3.13}$$

$$L = L_M + \lambda_C L_C + \lambda_F L_F \tag{3.14}$$

In practice, we use colors that have had gamma corrected back to linear intensity, and use a sigmoid to map from an unconstrained parameterization to $[0, 1]$. Flow is measured in the scale of half the shorter image dimension. This lets us set $\lambda_C = 4.5$ and $\lambda_F = 210$ for all of our experiments. We used identical settings for all our experiments: initialization is from a randomized small sphere of Gaussians, and we use fixed parameters for learning rate, canonical rescaling, batch size, and all other known parameters. Learning rates are automatically decayed based on statistical criteria [KH22].

We use the Adam [KB15] optimizer and rescale all scenes to a canonical size based on camera pose distances to balance the optimization of means and precisions [KH22] that occurs when using re-scaling optimizers. We randomly sample minibatches of 50,000 rays from across the entire sequence, which leads to extreme fast convergence, often getting a reasonable shape in well under an epoch (Figure 3.7). We use the per-Gaussian simple colors used in prior work [KH22] for simplicity, but extensions to Spherical Harmonics are possible for greater color fidelity per Gaussian [Fri+22; Ker+23].

Examples of our reconstructions can be seen Figure 3.2. Even though the optimization only takes minutes, operates on reasonably low resolution images, with reasonably few Gaussians, we can still see good results. Depth estimates are able

to capture small geometric details (notice the kickstand and both mirrors on the motorcycle). Despite the initialization being a small, invisible sphere of Gaussians, the optimization procedure is able to reconstruct shapes with rich geometry (such as the bicycle and the plant). Lastly, the color results look reasonably realistic. Although separate Gaussians must be used to paint 2D flat textures onto surfaces, reasonable results are obtained for the toy truck and the skateboard reconstructions. In the toy truck, grey side and roof panel details appear in the reconstruction. In the skateboard, the painted curve shape is also approximately modeled in the reconstruction, as are both wheels.

We obtained similarly good results with both Section 3.3.1 and Section 3.3.3, but all results in Figure 3.2 use the faster, two parameter model for optimization and the resulting visuals.

3.5 Reconstructing with Optical Flow

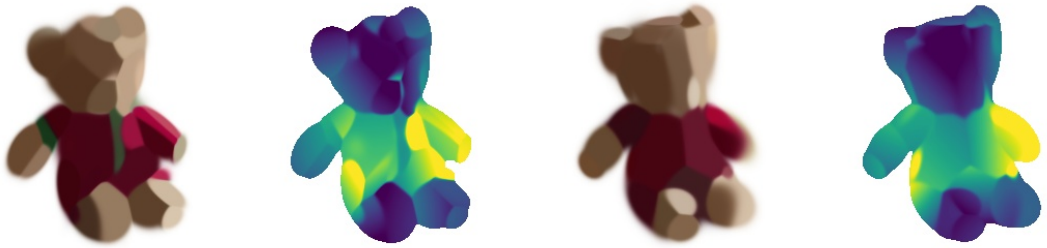
Many approaches to 3D reconstruction focus only on reconstructing the independent images from the given sequence [Ker+23], including all the 3D Gaussian methods [Ker+23; KH22; Wan+23]. However, in practice, these image sequences are often collected by cell phone videos [Rei+21] and have a strong temporal prior. Inspired by work in reconstructing 4D scenes [Li+21; Yan+21a; Yan+21b], we leverage optical flow in producing more precise 3D reconstructions of static objects.

Optical flow provides a hypothesis of surface correspondence, which regularizes the shape reconstruction. Correspondence can be essential in classic techniques for shape estimation [KRN97; LB81]. Sparse particle video trackers extend this, and obtain long-term video correspondences [HFF22].

Optical flow is a useful signal since it is a local estimator, and is robust to the long-term lighting changes that occur when typical users capture scenes under auto-exposure [Jun+22]. This makes it perhaps a more appropriate loss term than color models, which would be sensitive to lighting changes. Additionally, color often means texture, which implies high frequency texture details that can be difficult to reconstruct [Mil+20a], and perhaps efficient shape estimation can do without.

The benefits of flow can be from our experiments, qualitatively in Figure 3.4 and quantitatively in Table 3.1. One interesting result is that even classic optical

flow [Far03] provides helpful cues to the shape optimization, even though it is very noisy (see Figure 3.4a). Even better, state-of-the-art flow methods such as Unimatch [Xu+23] are extremely fast (real-time on GPU hardware) and have learned good priors even in texture-less areas. Using such learned flow maps (Figure 3.4b) can help shape estimation greatly. After optimization, predicted model flow (Figure 3.4d) is very similar to the given flow estimate from the network that was used to supervise it. Incorporating flow also produces significantly smoother depth maps, as can be seen in Fig. 3.3.



(a) **Without Flow** (b) **With Flow**

Figure 3.3: **Optical flow improves reconstruction**. We show the results of fitting a set of 3D Gaussians to a CO3D [Rei+21] sequence. Without a flow loss term, colors are estimated well but the shape is not. After adding flow, color fidelity is sacrificed but shape estimation improves.

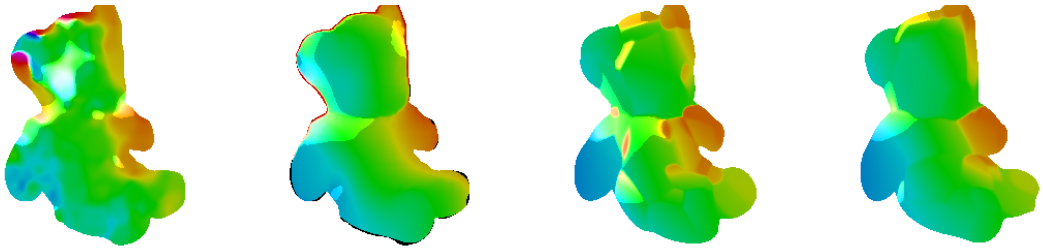
In our experiments, optical flow slightly hurts the color fidelity of the reconstructed models, but provides much more accurate shape reconstruction, as can be seen in Figure 3.3. After adding a flow loss term, this surrogate estimate of surface correspondence helps resolve concavity/convexity ambiguity from the silhouette and color loss terms. For example, the body of the teddy bear becomes smooth and its arms become well defined.

With the ray-based differentiable rendering of 3D Gaussians, it is reasonably easy to compute per-pixel optical flow. It only requires taking the 3D coordinate obtained from Equation (3.3), and transforming it with adjacent camera poses and projecting back into camera coordinates.

In our implementation, we pass all camera poses into the rendering function and represent the camera with a single inverse focal length parameter (making the

assumptions that the images lack distortion, the projection is in the center of the image, and that the pixels are square). After computing the depth image for a given frame, we transform the point cloud (forwards and backwards) and project the transformed coordinates into the image. The changes in coordinates is the direct, per-ray optical flow estimate. Without using another source to regularize the optical flow, we get reasonable estimates but with some artifacts due to shape uncertainty (as can be see in Figure 3.4c).

We include estimates for both forward flow (pose i to pose $i+1$) and backward flow (pose i to pose $i-1$) from our differentiable renderer, for each ray.



(a) Gunnar [Far03] (b) Uni [Xu+23] (c) Without (d) With

Figure 3.4: **Optical Flow Quality**. (a) shows a classic optical flow estimate. (b) shows a modern, learned optical flow estimate. (c) and (d) show the estimated flow for the same frame after fitting 3D Gaussian model; respectively without adding a flow loss and after adding a flow loss term to fit the estimate from (a). Using standard optical flow coloring [Bak+07].

Figure 3.5: **Colored Mesh Exports** reconstructed from CO3D sequences [Rei+21] with 40 3D Gaussians and exported to Blender (Section 3.6).

	Depth Error	**Runtime (seconds)**
No Color or Flow	0.271	17
Color	0.262	15
Color & Classic Flow [Far03]	0.237	**14**
Color & Learned Flow [Xu+23]	**0.155**	15

Table 3.1: **Optical flow helps reconstruction** of CO3D sequences [Rei+21]. For details see Section 3.5.

3.6 Exporting Meshes

Many successful differentiable renderers realize that fast, efficient, robust differentiable rendering often require smooth, fuzzy and indefinite surface representations [Mil+20a; KH22; Ker+23]. For optimizing shapes from videos, it is helpful to have some degree of softness to aid gradient flow.

On the other hand, commercial rasterization pipelines in most desktop and mobile GPUs typically operate on triangle meshes [Guo+23; Che+23]. Additional, the field of shape processing often prefers not just definite surfaces in the form of meshes, but watertight meshes [ZJ16; Hu+22]. Differentiable mesh renderers typically bridge this divide with an explicit spatial smoothness term [Liu+19; Rav+20].

Instead, 3D Gaussians, neural surfaces, and other similar methods are implicit surface methods, where the genus of the object can change during optimization. Many existing works export meshes using marching cubes [LC87; Mil+20a; KH22], where a volumetric grid is evaluated and triangles are produced at edges crossing a particular level set threshold. While fast and simple, this can produce non-watertight meshes. Additionally, it can require searching over an ideal threshold to find which level set of the implicit surface matches the explicit surface best [KH22].

In this work, we leverage the surface definition used by the underlying differentiable renderer, and then solve the appropriate Poisson equation [KBH06; KH13]. In Poisson surface reconstruction, oriented point sets (points with normals indicating the local

tangent plane of the surface) are used as input to solve a Poisson equation:

$$\nabla \cdot \nabla \chi = \nabla \cdot \vec{V}. \tag{3.15}$$

Solving these equations can be done via well-conditioned sparse linear systems [KBH06]. Each point in the oriented point cloud provides an estimate of the local gradient of the indicator function (which points are inside the surface of the object). A nice property of Poisson Surface solvers is that their solutions always produce watertight meshes, as they solve for an indicator volume, which produces a surface that is a $\mathbb{R}^2$ manifold folded in $\mathbb{R}^3$. This process can be done over a basis function set of B-splines with compact support.

To produce an oriented point set, we simply go over all of our training views and render the point cloud of the object, to produce point locations $p = (x, y, z)$. The orientation of each point can be produced in two different ways (shown in Section 3.6). The first, and most general, is to use the rasterization locality by taking a horizontal (p_x) and a vertical (p_y) screen-space neighbor for each point (p) and taking their cross product:

$$\vec{n} = \frac{(p - p_x) \times (p - p_y)}{||(p - p_x) \times (p - p_y)||}. \tag{3.16}$$

This technique works for any differentiable renderer producing images on a grid (hence rasterization), but produces poor results on discontinuities.

An alternative approach for 3D Gaussians is to re-use Equation (3.3) to blend all the local estimates of the normal ($\vec{n_i}$ instead of t_i) into a final estimate. The local estimate of the normal is given by the derivative of Equation (3.1).

$$\vec{n_i} = \frac{\Sigma^{-1}(vt_i - \mu_i)}{||\Sigma^{-1}(vt_i - \mu_i)||}. \tag{3.17}$$

In general, the sign and scale of the normal is fixed: the sign of the normal should face the camera created it, and the $||\vec{n_i}|| = ||\vec{n_f}|| = 1$. Both of these techniques can be seen in Section 3.6. In practice, the blended definition is preferred as it requires no neighbors.

For producing meshes from our renderer, we typically perform the faster optimization with Section 3.3.1. and the rendering equation from Section 3.3.3 can directly

render that representation with no changes. The alpha compositing definition is preferred for mesh exporting as it is more view consistent than the heuristic blending. Lastly, we can reject any intersections that fail a quality threshold of a direct intersection, $\max_i(w_i) \leq \epsilon$ which is typically set to 0.9. The Poisson surface reconstruction produces a surface interpolation, and so the sparsity of point samples is not a problem. Additionally, we can export a colored, oriented point cloud, where the color is either the reconstructed color or the color of the images themselves at those points (see Figure 3.5). We obtained object reconstructions by solving Equation (3.15) with Dirichlet boundary constraints [KH13].

An intuition for why we can simply export strong hits is that Gaussians fit to data are typically not overlapping. For example, for a GMM, consider evaluating how independent each Gaussian is from each other:

$$r(i) = \frac{\sum_{j;j\neq i} \lambda_j p_j(\mu_i)}{\lambda_i p_i(\mu_i)}$$

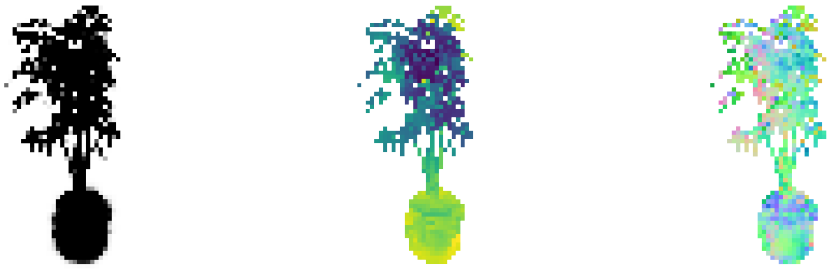
In our experiments, $r(i)$ tends to average less than 5% for Gaussians fit to point clouds with EM. This suggests that the shapes are largely overlapping.

3.7 Interoperability

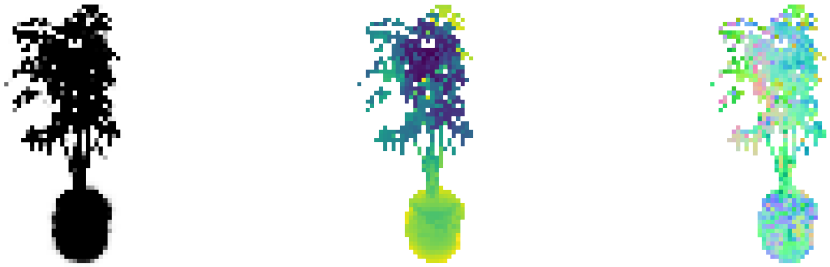
We describe how Fuzzy Metaballs [KH22] and 3D Gaussian Splatting [Ker+23] share a similar underlying shape representation. We demonstrate this by showing results in Figures 3.6 and 3.9 where the initial shape reconstruction was performed using the 3D Gaussian Splatting paper [Ker+23], and then directly converted and rendered using the methods in Sections 3.3.1 and 3.3.3 and then exported using the oriented points using Section 3.6.

Our experiments in Figure 3.7 and Section 3.6 show that Sections 3.3.1 and 3.3.3 use mutually compatible definitions of shape representation. However, the Gaussian Splatting [Ker+23] work uses an entirely different code base, in a different framework, optimizing scenes instead of objects, with no object masks, with a custom CUDA kernel, and preferring α to δ estimates. However, since both are using 3D Gaussians, we can render one with the other.

We convert the Gaussian Splatting method to be compatible with our approach

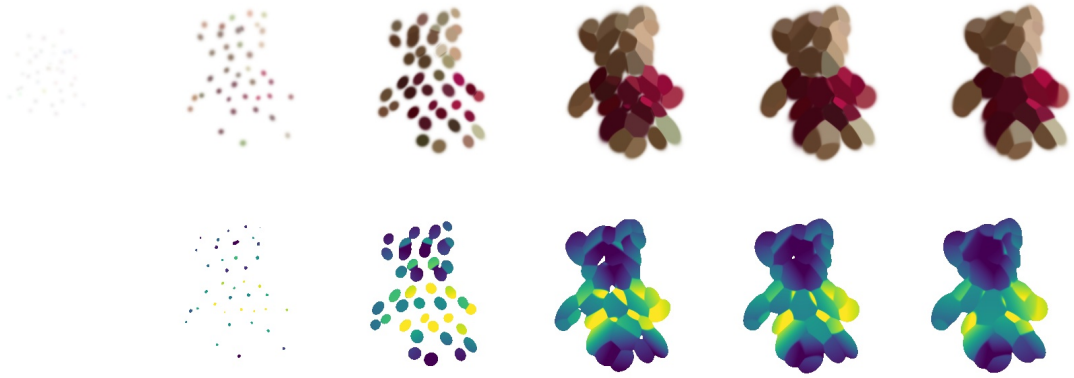


(a) **Ficus via Weighted Blending (Section 3.3.1)**

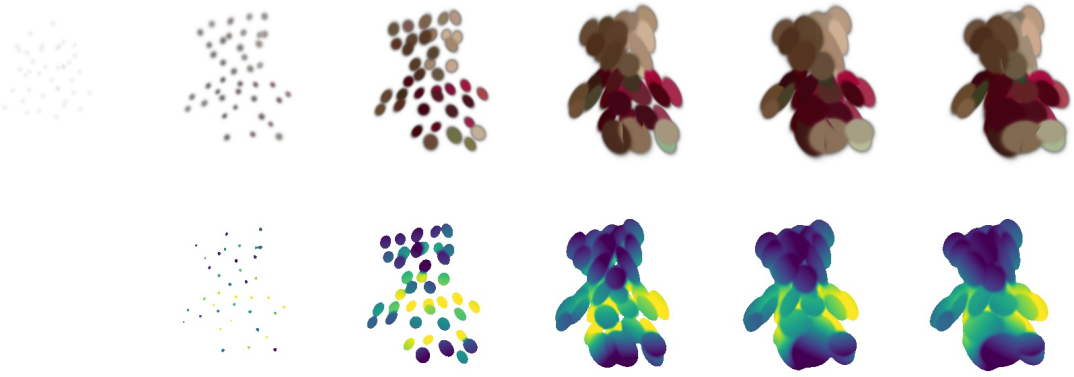


(b) **Ficus via Alpha Compositing (Section 3.3.3)**

Figure 3.6: **3D Gaussians are Fuzzy Metaballs** Reconstructed ficus from Gaussian Splatting [Ker+23], rendered with Fuzzy Metaballs [KH22]. Shown are opacity, depth & normals.



(a) **Optimization via Weighted Blending (Section 3.3.1)**



(b) **Optimization via Alpha Compositing (Section 3.3.3)**

Figure 3.7: **Both forms of rendering behave similarly.** Shown are 1%, 4%, 7%, 12%, 16% and 20% of the first epoch.

Figure 3.8: **Visualization of Normals** for 3D Gaussians.

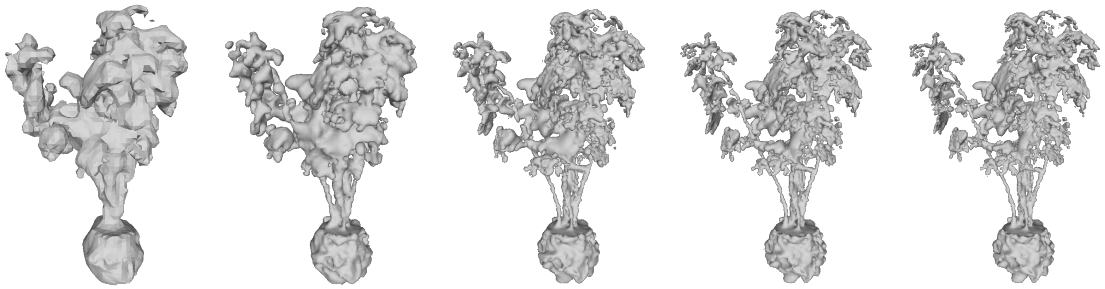


Figure 3.9: **Visualization of Mesh Export** This ficus shape was reconstructed using the Gaussian Splatting [Ker+23] code but rendered as an oriented point cloud with Fuzzy Metaballs [KH22] and reconstructed with a Poisson Solver [KBH06; KH13]. Left to right are Poisson tree depths of 6 through 10. While deeper tree depths produce more detailed reconstruction, noise and artifacts are also amplified. The mesh export is based on an oriented point cloud exported from 100 images of size 80×60. It is interesting to see that precise details like the various steps become visible, even when the forward passes were low resolution (see Figure 3.6 for a visual example of the coarseness generated by the renderer).

with only a few steps. Means are used directly, each Σ are converted to $\Sigma^{-\frac{1}{2}}$, and the α for each Gaussian is replaced. For simplicity, we ignore $\alpha_i < 0.5$ and set $\lambda_i = \log(80)$ for the remaining 3D Gaussians, which we found works reasonably well[1]. About 90% of Gaussians had insufficient α, creating a ten times speedup in our experiments. For weighted blending experiments, we reused the settings of $\beta_1 = 21.4$ and $\beta_2 = 3.14$ from our prior experiments.

As can be seen in Figure 3.6, both techniques are able to capture the fine stem and leaf structures in the reconstructed ficus plant. The weighted blending technique demonstrates smoother normals, but the mesh export uses the alpha composting method and reasonable meshes can be obtained.

In Figure 3.9 we show Poisson surface reconstructions of the ficus, where the oriented point cloud used for solving the equation was generated by our renderer, but the original reconstruction was made with 3D Gaussian splatting. We show reconstructions at different tree depths, showing an increase in details, and an increase in noise, at finer scales. The solver can recover reasonably fine details

[1]Inverse sigmoid conversion of $\lambda = -C\log(1-\alpha)$ maybe also be reasonable with appropriate C, which we did not search for.

considering considering the low resolution of the oriented point clouds.

This extends the utility of the 3D Gaussian Splatting approach to be rendered with our fast methods that have viable JAX [Bra+18] CPU and GPU backends. Our approach enables per-ray depth computation, normals and mesh exporting (Section 3.6). The ficus data was provided without color, so we show colored exports in Figure 3.5. Mesh exports provides an interconnect with most 3D creation tools.

It remains to be seen which approach produces better optimization for shape reconstruction. The weighted blending FM approach allows two Gaussians to be blended together and averaged, but 3DGS and the sorting method both require explicit occlusion of Gaussians based on z-order, perhaps harming the smoothness of the optimization process.

	CPU	**GPU**
Weighted Blending (Section 3.3.1)	4.94 μs	226.6 ns
Alpha Compositing (Section 3.3.3)	12.7 μs	377.6 ns

Table 3.2: **Runtime per ray, for an entire iteration** with a i5-7267U CPU and a GTX 1080 GPU. This was done with a 40 Gaussian model and these times include memory transfer times and forwards and backwards passes. Alpha compositing drops the need for hyper-parameters, in exchange for 200% slower runtimes on CPU and 50% slower runtimes on GPU. Since both approaches behave similarly (Fig. 3.7) and are interoperable (Section 3.6), this creates a trade-off between performance and simplicity.

3.8 Splitting Gaussians

While the 3D Gaussian Splatting [Ker+23] explores some heuristic techniques for merging and splitting of Gaussians, here we develop an alternative, deterministic approach to modifying the number of Gaussians in the reconstruction.

We split 3D Gaussians after our initial model has converged according to statistical criteria [KH22]. And we then repeat the optimization process. Two steps of this are shown in Figure 3.10. The splitting and removal process is based on removing

Figure 3.10: **Re-parameterized 3D Gaussians** to minimize apparent loss. N is the number of mixtures used. The appearance of the tie and the sleeve in green can be seen with added Gaussians.

Gaussians that contribute minimally to the reconstruction and splitting Gaussians that are given too much reconstruction responsibility based on the chosen loss.

To compute the set of Gaussians that minimally contribute, we compute the means and standard deviations of the weights assigned to each Gaussian μ_λ and σ_λ and remove Gaussians that satisfy

$$\lambda_i \leq \mu_\lambda - z_\lambda \sigma_\lambda, \tag{3.18}$$

where z_λ is a z-score typically set to 2.

To compute the splitting criteria, we take a random batch of rays (typically about 5% of the dataset) and compute the per-ray loss (e.g. Eq. (3.14) but any reconstruction loss is viable). Each ray also has an associated set of computed weights w_i (Eqs. (3.7) and (3.9)). Over this batch of M rays, we estimate the average loss associated with each Gaussian as $\bar{l_i} = \frac{1}{M}\sum_j^M = L_j \cdot w_i$. We compute the means and standard deviations of these losses as $\mu_{\bar{l}}$ and $\sigma_{\bar{l}}$ and split Gaussians that satisfy

$$\bar{l_i} \geq \mu_{\bar{l}} + z_{\bar{l}}\sigma_{\bar{l}}, \tag{3.19}$$

where $z_{\bar{l}}$ is a z-score typically set to 1.

Gaussians are split by forming two Gaussians by using the properties of the

half-normal distribution [Dan59]. The two new means are generated by shifting the initial mean (μ_i) in opposite directions by the direction of the dominant eigenvector ($\vec{v_1}$), and by the appropriate factor of the dominant eigenvalue (σ_1) of the covariance matrix (Σ_i):

$$\mu_{a,b} = \mu \pm \sigma_1 \vec{v_1} \sqrt{\frac{2}{\pi}}. \tag{3.20}$$

Both new Gaussians are given identical covariance matrices, reconstructed from the initial eigenvalues and eigenvectors but with a scaled dominant eigenvalue:

$$\sigma_{1_{a,b}} = \sigma_1 \sqrt{1 - \frac{2}{\pi}}. \tag{3.21}$$

The scaling factors $\sqrt{\frac{2}{\pi}} \approx 0.8$ and $\sqrt{1 - \frac{2}{\pi}} \approx 0.6$ are based on the properties of the half-normal distribution.

We replicate the initial weights (λ) and colors (c_i) for the new Gaussians, with noise ($\epsilon_c = \epsilon_w = 0.1$) in their unconstrained parameterization space to avoid issues with coupled gradients during further optimization:

$$\log(\lambda_{a,b}) = \log(\lambda) + \mathcal{N}(0, \epsilon_w), \tag{3.22}$$

$$\sigma^{-1}(c_{a,b}) = \sigma^{-1}(c) + \mathcal{N}(0, \epsilon_c). \tag{3.23}$$

This approach to splitting is deterministic and allows for increasing detail in an iterative way, as shown in Fig. 3.10.

3.9 Discussion

With the increased deployment of vision systems in everyday environments, there is a need for flexible, efficient, and easily computed shape reconstructions. The recent developments in photorealistic differentiable rendering make close the dream of virtual systems accurately capturing everyday objects in the virtual world. 3D Gaussians, or metaballs, are a simple and powerful representation for shapes that enables easy reconstruction, as has been shown by prior work [Ker+23; KH22; Wan+23]. These techniques extend and interconnect these approaches and provide them additional flexibility.

The equivalence of weighted blending and alpha composting approaches provides a wider array of options — one being significantly faster and the other lacking hyper-parameters. The computation of per-ray blended normals allows for reliable mesh exporting, connecting to other 3D techniques and methods, without the need to pick thresholds for marching cubes [LC87], and producing watertight meshes via Poisson reconstruction [KH13]. We show a more deterministic, grounded approach to performing reparameterization of Gaussians. Lastly, optical flow as a regularizer and prior for surface correspondences can easily and reliably improve the quality of shape reconstructions.

3.10 Conclusion

We have extended existing approaches for differentiable rendering of 3D Gaussians for speed, simplicity and flexibility. This expanded flexibility should allow these representations to be used in more places, for more applications, and potentially across a wider array of computational platforms than before.

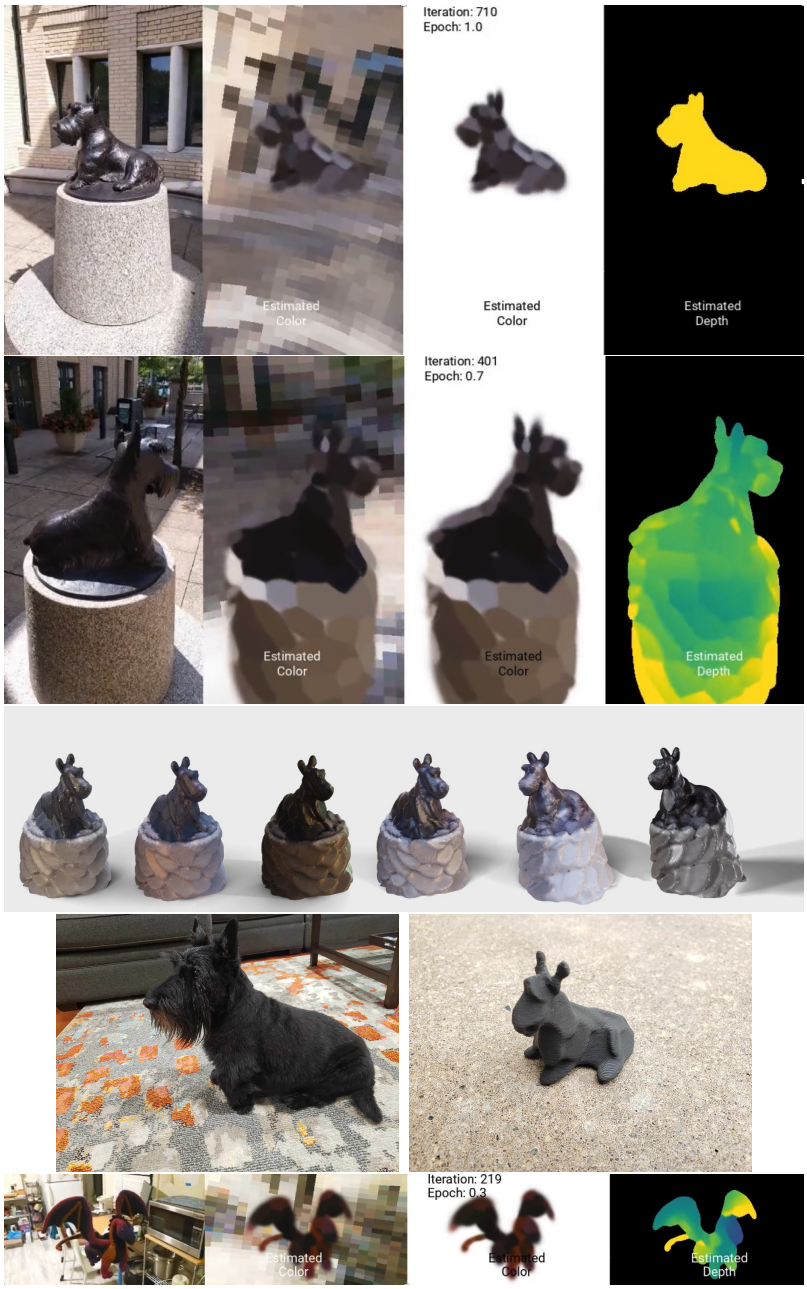


Figure 3.11: **Reconstruction of Scottish Terrier** Statue on the Carnegie Mellon Campus. Performed by running SAM [Kir+23] on the first frame and propagating with XMem [CS22], before performing the optimization described in this section. Different reconstructions with different masks are possible, as is relighting the reconstructed meshes in Blender. Charlie, my actual Scottish Terrier, is shown for reference. As is a printed 3D model of 32 Gaussian reconstruction of the object. And a dragon (crocheted by K. Shih) reconstruction to represent CMU SCS.

Chapter 4

Direct Fitting of Gaussian Mixture Models to Meshes

In this chapter, we present a formulation for fitting Gaussian Mixture Models (GMMs) directly to a triangular mesh instead of using points sampled from its surface. This is largely similar to our published manuscript on this topic [KH19].

Part of this work analyzes a general formulation for evaluating likelihood of geometric objects. This modification enables fitting higher-quality GMMs under a wider range of initialization conditions. Additionally, models obtained from this fitting method are shown to produce an improvement in 3D registration for both meshes and RGB-D frames. This result is general and applicable to arbitrary geometric objects, including representing uncertainty from sensor measurements.

4.1 Introduction

In robotics and computer vision, there exist many forms of representation for 3D geometric data. For example, some researchers use unordered point sets [Pau92], others require points with surface normals [KBH06] or dense volumetric representations such as signed distance fields [New+11]. The variation in forms of representation is related to the wide variety of sources and uses for this data, from raw depth sensor measurements to Computed-Aided Design (CAD) models.

Many researchers have found that Gaussian Mixture Models provide a power-

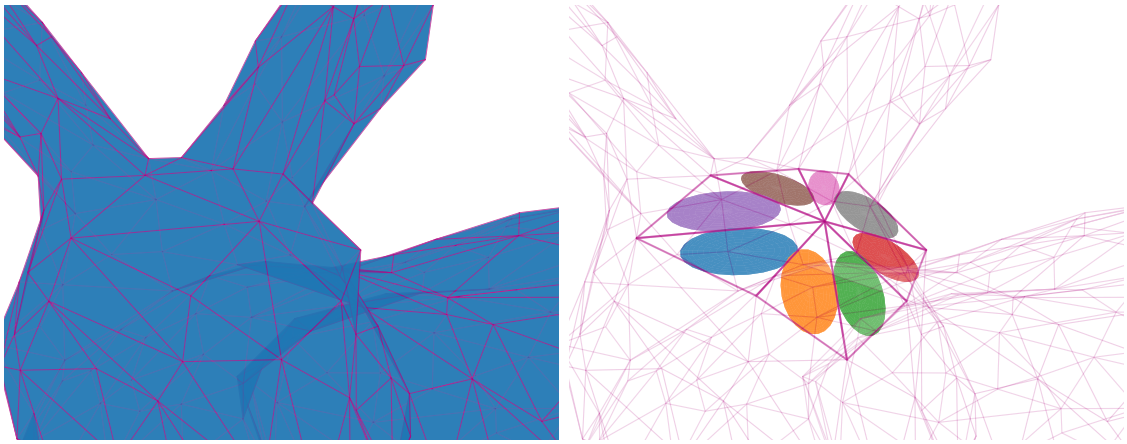


Figure 4.1: **Visual example of the Stanford Bunny**, highlighting 8 triangles on the head. Each triangle is shown with its covariance (plotted to 1.5σ). We demonstrate how Gaussian Mixture Models can use the covariance information from given geometric structures (in this case, triangles) to fit models more efficiently, robustly and with higher fidelity.

ful representation, especially for use in registration between unknown poses [JV11; Eck+15; Eck+16; EKK18; TOM18]. Producing a Gaussian Mixture Model requires only unstructured point sets from the underlying geometric data, which makes them widely applicable. Our contribution in this work is to demonstrate how Gaussian Mixture Model can be constructed, evaluated on and fit directly to geometric primitives, such as the triangles of a polygon mesh. This is done by incorporating the structural information from each primitive into the algorithm, for a visual example, see Figure 4.1.

Our contributions include a mathematical framework for how geometric primitives can be incorporated with probability distributions (Section 4.2.2). We demonstrate how to obtain the structural properties for a triangular mesh (Section 4.2) and how it can be generalized to other primitives (Section 4.5.1). Incorporating structural information allows us to build Gaussian Mixture Models that not only converge faster and in more conditions (Section 4.4) but also provide a representation that produces higher quality registration results when the models are used in practice (Section 4.6).

The code for all methods and experiments in this chapter is available at `https://github.com/leonidk/direct_gmm`.

4.2 Method

4.2.1 Gaussian Mixture Models

The Gaussian Mixture Model (GMMs) is a well studied probability distribution. It is possible to fit these models to empirical data via Maximum Likelihood Estimation (MLE) [Has66; DLR77]. The likelihood of any point x in a Gaussian is given by

$$\mathcal{N}(x;\mu,\Sigma) = 2\pi^{-k/2}\det(\Sigma)^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{4.1}$$

where $\mu \in \mathbb{R}^k$ is the mean and $\Sigma \in \mathbb{R}^{k\times k}$ is the positive-definite covariance matrix. The log-likelihood of a given point x is given by

$$\begin{aligned}\log\mathcal{N}(x;\mu,\Sigma) = &-\frac{k}{2}\log(2\pi)\\ &-\frac{1}{2}\log(\det(\Sigma))\\ &-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\end{aligned} \tag{4.2}$$

In a Gaussian Mixture Model with K components, with λ_i are mixing weights subject to $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0\ \forall i$, the probability of a point x is given by

$$g(x) = \sum_{i=1}^{K} \lambda_i\ \mathcal{N}(x;\mu_i,\Sigma_i)$$

4.2.2 Geometric Objects in a Probability Distribution

First we must handle the general case of how to evaluate the probability of a known geometric object in a probability distribution. Consider sampling N points from the geometric object, where a notion of likelihood can be evaluated by taking their product. To account for the variable number of samples, we take a geometric mean. And to obtain the likelihood of the object, we take the limit as the number of samples

grows to infinity.

$$
\begin{aligned}
\ell &\cong \prod_{i=1}^{N} p(x_i) \\
\ell &\cong \left(\prod_{i=1}^{N} p(x_i) \right)^{\frac{1}{N}} \\
\ell &= \lim_{N \to \infty} \left(\prod_{i=1}^{N} p(x_i) \right)^{\frac{1}{N}} \\
\ell &= \lim_{N \to \infty} \exp\left(\log\left(\left(\prod_{i=1}^{N} p(x_i) \right)^{\frac{1}{N}} \right) \right) \\
\ell &= \lim_{N \to \infty} \exp\left(\frac{1}{N} \sum_{i=1}^{N} \log(p(x_i)) \right) \\
\ell &= \exp\left(\int \log(p(x)) dx \right)
\end{aligned}
\tag{4.3}
$$

Equation 4.3 is a form of product integral [Fey51; Gue83; BKÖ08], which can be used to evaluate the likelihood of a known geometric object. The extension to multiple objects is straightforward, simply adjust their sampling weights accordingly. This form of product integral has two nice properties, it is invariant to resampling and it produces a result of 0 if $p(x) = 0$ anywhere along the geometric object. By invariance to resampling, we mean that one large primitive with sophisticated integration bounds gives the same answer as many small disjoint pieces of surface with their own bounds.

Following the use of Jensen's inequality, we get the following lower bound on likelihood

$$
\begin{aligned}
L &= \exp\left(\sum_{j=1}^{M} \int_{\triangle} \log\left(\sum_{i=1}^{K} \lambda_i \, \mathcal{N}(x; \mu_i, \Sigma_i) \right) dx \right) \\
\log(L) &= \sum_{j=1}^{M} \int_{\triangle} \log\left(\sum_{i=1}^{K} \lambda_i \, \mathcal{N}(x; \mu_i, \Sigma_i) \right) dx \\
&\geq \sum_{j=1}^{M} \sum_{i=1}^{K} \int_{\triangle} \log\left(\lambda_i \, \mathcal{N}(x; \mu_i, \Sigma_i) \right) dx
\end{aligned}
\tag{4.4}
$$

We also consider a simplified model, where each triangle is sampled at its center of mass (μ_j), and has weight corresponding to its area (α_j). As combining probabilities is done with multiplication, we use a weighted geometric mean over all points, obtaining the following approximation

$$L \approx L_S = \prod_{j=1}^{M} \left(\sum_{i=1}^{K} \lambda_i \, \mathcal{N}(\mu_j; \mu_i, \Sigma_i) \right)^{\frac{\alpha_j}{\sum_k \alpha_k}} \tag{4.5}$$

4.3 Modifying EM maximization to account for triangles

In this section we derive the expressions for fitting a GMM to a triangular mesh. We will represent each GMM component with parameters $\mu_i, \Sigma_i, \lambda_i$ and each triangle with vertices A_j, B_j, C_j, centroid μ_j and area α_j. Traditional EM minimization is possible by analyzing the lower bound of the log-likelihood. Following standard formulations (see [DLR77; Jor09; Sri14]), we add mixture sampling probabilities η_{ij} and move the logarithm inside the summation to obtain a valid lower-bound to minimize by using Jensen's inequality.

To perform fitting, we need an M-step which obtains $\lambda_i, \mu_i, \Sigma_i$ by maximizing the lower bound

$$LB = \sum_{j=1}^{M} \sum_{i=1}^{K} \eta_{ij} \log(\lambda_i \mathcal{N}(x_{jk}; \mu_i, \Sigma_i)) \tag{4.6}$$

To maximize this expression, we can take derivatives with respect to the variables of interest and obtain

$$\begin{aligned} \frac{\partial LB}{\partial \mu_i} &= \frac{1}{2} \sum_{j=1}^{M} \Sigma_i^{-1} (x_j - u_i) \eta_{ij} \\ \frac{\partial LB}{\partial \Sigma_i^{-1}} &= \frac{1}{2} \sum_{j=1}^{M} \eta_{ij} \left(\Sigma_i - (x_j - \mu_i)(x_j - \mu_i)^T \right) \end{aligned} \tag{4.7}$$

Now we can integrate these three expressions over the two dimensional surface of a triangle via a change of variables substitution, then set the result equal to zero and

solve, thus obtaining the update equations. For clarity we will also define a weight variable w_{ij} and its corresponding normalization constant W_i

$$w_{ij} = \eta_{ij}\alpha_j$$

$$W_i = \sum_{j=1}^{M} w_{ij}$$

The resulting lower-bound likelihood can be written as

$$\begin{aligned}\log(L) \geq \frac{1}{2}\sum_{j=1}^{M}\sum_{i=1}^{K} w_{ij} \\ &[2\log(\lambda_i) - k\log(2\pi) - \log(\det(\Sigma_i)) \\ &- (\mu_j - \mu_i)^T\Sigma_i^{-1}(\mu_j - \mu_i) \\ &- \frac{1}{12}(A_j^T\Sigma_i^{-1}A_j + B_j^T\Sigma_i^{-1}B_j + C_j^T\Sigma_i^{-1}C_j \\ &-3\mu_j^T\Sigma_i^{-1}\mu_j)]\end{aligned} \tag{4.8}$$

The new mean is obtained as simply weighted mean of centroids. This update equation is identical to the one derived for the approximation in equation 4.5

$$\mu_i = \frac{1}{W_i}\sum_{j=1}^{M} w_{ij}\mu_j \tag{4.9}$$

The same technique will provide an answer to the update equation for covariance.

$$\begin{aligned}&\Sigma_i \\ &= \frac{1}{W_i}\sum_{j=1}^{M} w_{ij}\left[\underbrace{(\mu_j - \mu_i)(\mu_j - \mu_i)^T}_{cov(\mu_j,\mu_i)} + \underbrace{\frac{1}{12}(A_jA_j^T + B_jB_j^T + C_jC_j^T - 3\mu_j\mu_j^T)}_{cov(\triangle_j)}\right]\end{aligned} \tag{4.10}$$

The final update is surprisingly simple, it is the area weighted average of the covariance obtained by using centroids as point measurements plus the covariance equation for a triangle. That is, at every iteration, each mixture is updated with some fraction of the structure of the triangles associated with it. A visual example of ellipsoids

showing triangle covariance structures is shown in Fig. 4.1. Our derived expression for the covariance of a triangle is expressed in terms of vertices, but is consistent with the standard formulation in CGAL [GAP08].[1] The update equation for eq. 4.5 is similar, simply lacking the $cov(\triangle_j)$ term.

4.3.1 Evaluating the derived loss function

To evaluate the validity of the expression in equation 4.8, we compare its fitting fidelity numerically against a large number of sampled points. The results are shown in figure 4.2. We can see that, the lower bound expression for triangles is equal to that obtained numerically from a large number of points. Since the lower-bound expression is all that's needed in EM optimization [DLR77], the equality of this expression suggests we can use it in fitting real data.

4.4 Results

We performed experiments fitting Gaussian Mixture Models to triangular meshes. We swept a wide range of K mixture components (from 6 to 400) and evaluated two different initialization schemes. The first initialization performs k-means++ [AV07] clustering, and uses those clusters as initial assignments for the EM method. The second method uses simple random assignments for initialization. We run 25 iterations of EM for all methods, with a tight tolerance ($\epsilon = 10^{-12}$) to prevent an early exit from the optimization.

To evaluate the converged model, we use a densely sampled point cloud of the initial mesh (Figure 4.5e) and report its likelihood according to equation 4.1. Since all of our experiments tend to operate on 1,000 points or triangles, the use of 50,000 points for evaluating the model should provide a good test of GMM model fidelity. In all of these cases, we focus on the Stanford Bunny. Visual examples of our input and output data is shown in Figure 4.5.

[1]The reference document has a typographic error in the moment matrix for triangles, which should be a 5x multiple of the one for 3D tetrahedrons. The CGAL source code correctly implements this matrix in practice.

4.4.1 Mesh Input Data

The first set of experiments, shown in Figure 4.3, compares fitting GMM models to different input formats of the *res4* Stanford Bunny. We compare our exact (eq. 4.8) and approximate (eq. 4.5) mesh loss equations against fitting a traditional point-loss to the triangle centroids and the mesh vertices. The best results came from our exact mesh expression, with the second best being its approximation. The proposed methods handled random initialization and k-means initialization. On the other hand, the point-based methods often had a preferred initialization. A qualitative look at the resulting models, shows that the mesh GMM (Fig. 4.5h) produces a fuller model of the Stanford Bunny than the center-of-mass GMM (Fig. 4.5d), even when the evaluated likelihood was numerically very similar.

4.4.2 Mesh Decimation

While the above experiments used the low-resolution *res4* Stanford Bunny, we also evaluated which GMM fitting strategy works best when subsampling high-resolution mesh data. In our case, we try two methods point-sampling (Poisson and Random) and two methods of triangle collapse (Quadric and Clustering). Our experimental procedure involves fitting a GMM to a low-resolution mesh (1,000 faces) or point cloud (1,000 points) and evaluating the likelihood of the resulting GMM against a dense point sampling of the original shape (50,000 points). The different sampling strategies can be seen visually in Figure 4.5.

To generate a low-resolution mesh, we try two methods, clustering decimation [RB93], which is fast, and quadric error decimation [GH97], which is more accurate. To generate a low-resolution point cloud, we both randomly sample points on the surface on the mesh, and use Poisson Disc sampling to ensure uniform samples [CCS12]. As before, the first is faster while the latter produces better results. Poisson Disc sampling is also used to generate the high-resolution, "ground truth", point cloud used for evaluation.

The results of these experiments are reported in Figure 4.4. As before, the mesh-based registration strategies are largely invariant to initialization method while the point-based strategies often prefer k-means initialization (exceptions discussed in Sec. 4.4.3). The best results often came from the use of Poisson Disc Sampling (with

k-means), which ensures uniform coverage of the surface areas. In contrast, using random samples generated the lowest quality results. The mesh-based techniques proved to be reliable across all tests, regardless of mixture number and initialization.

4.4.3 Discussion

All of these experiments were run using the *GaussianMixture* code base from `scikit-learn v0.20.0` [Ped+11]. We modified the code to support additional weight and covariance terms (Sec. 4.5.1), which were general enough to allow us to implement all proposed methods.

One surprising result in Figures 4.3 and 4.4 was that random sampled points performed better with random initialization than k-means initialization (at high mixture numbers). As the EM algorithm only finds a local minima, this suggests that k-means may not always be an ideal initialization technique. We believe that this occurred due to either a bad local minima from initialization, or fairly flat cost during optimization, leading to an early exit condition being triggered (despite our tight tolerance of $\epsilon = 10^{-12}$). This behavior was never observed when using our proposed exact mesh formulation.

We used 25 iterations for all of these experiments. When 100 iterations were used, the point-based methods performed better (nearly as good as our proposed method). However, our method often converged in about $\frac{1}{3}$ the number of iterations, so we picked a lower iteration number for consistency in runtime.

4.5 Extensions

4.5.1 Generalization to other primitives

While equations 4.8,4.9,4.10 were derived specifically for triangles, the update equations can be written more generally for any primitive (triangles, Gaussian mixtures, cuboids, etc.) using $\mu_p, \Sigma_p, \alpha_p$ to denote primitive's mean, covariance and size ($\int_S dS$) respectively. Then the loss, mean update, and covariance update equations for a Gaussian Mixture can be written with equations 4.11,4.12,4.13. The previous equations can be seen as a special case of these formulas, which provide an M-step update

for any set of geometric primitives $p \in \mathbf{P}$ in fitting a Gaussian Mixture Model.

$$\log(L) \geq \frac{1}{2} \sum_{p=1}^{P} \sum_{i=1}^{K} w_{ip} \left[2 \log(\lambda_i) - k \log(2\pi) \right. \\ \left. - \log(\det(\Sigma_i)) - (\mu_p - \mu_i)^T \Sigma_i^{-1} (\mu_p - \mu_i) - \Sigma_p \right] \tag{4.11}$$

$$\mu_i = \frac{1}{W_i} \sum_{p}^{P} w_{ip} \mu_p \tag{4.12}$$

$$\Sigma_i = \frac{1}{W_i} \sum_{p}^{P} w_{ip} \left[(\mu_p - \mu_i)(\mu_p - \mu_i)^T + \Sigma_p \right] \tag{4.13}$$

We note that these are the exact same update equations used in fitting hierarchical Gaussian Mixture Models [VL99]. However, while previous work applied these equations to fitting GMMs to existing GMMs, we show how this update can be used for fitting geometric data. This general form allows for easy substitution of known structural information (in the form of a second moment) about any geometric primitive. Computing covariance structures for arbitrary polyhedra is well studied area of research [Mir96; GAP08].

4.5.2 Number of Mixtures

In our later experiments, we fix the number of mixture models. Unless otherwise stated, we use $K = 100$. A visual example of this mixture can be see in Figures 4.5d and 4.5h. We picked $K = 100$ as this matches the experimental conditions recommended for using GMMs for SLAM [TOM18]. In practice, there are many ways to select this number, including flatness of the distribution's KL-divergence [SM16; DSM18], flatness of the mixture themselves [EKK18], or by evaluating an information criterion [PM00]. We believe that when this technique is used in practice, this number can either be found through cross-validation [TOM18] on a registration dataset or by using external system information such as depth sensor noise models [Kes+17].

4.6 Applications

The experiments in section 4.4 showed that fitting Gaussian Mixture Models using structural information tends to produce higher quality probability distributions. Some recent work has focused solely on the efficient nature of GMMs in representing shapes [Eck+16]. Here we show that our improvements in model quality produce an appreciable performance improvement in actual 3D computer vision applications.

Gaussian Mixture Models have found wide use in the 3D registration literature. From the Normal Distance Transform [Mag+09; Sto+12], to variants of the L_2 loss [JV11; SML12] and even Coherent Point Drift [MS10], many 3D registration methods utilize Gaussian Mixture Models. Their benefits include robustness to noise, smooth variation over 3D space, speed of evaluation, and straightforward control over model complexity. These models can provide results that are state-of-the-art in both runtime and registration accuracy [EKK18]. We show that applying our proposed mesh GMM fitting can produce an improvement in these results.

4.6.1 Mesh Registration

We replicate the experimental setup of a recent paper [EKK18], demonstrating how Gaussian Mixture Models can be used for efficient 3D registration. As our experimental setup matches [EKK18], the 20 different dozen registration methods compared in Figure 3 of that work can be directly compared against the results here. Their experiment operates on taking a large number of random deformations of the Stanford Bunny and evaluating the final quality of fitting result. Our results can be seen in Figure 4.6.

To perform 3D registration, we first build a GMM for the *res4* Stanford Bunny, as in Section 4.4.1, using 100 iterations of EM with a tolerance of 10^{-5}. We then sample vertex number of 3D points from the surface of the mesh ($N = 453$). While previous work has focused on a Point-to-Distribution (P2D) technique with a polynomial approximation of the likelihood function [Mag09], we do straightforward P2D in our experiments. Our registration process consists of finding the rigid body transformation that maximizes equation 4.1. We use the identity transformation for initialization and then perform gradient-based optimization to find the local minima.

We compute gradients using numerical differences. For the optimizer, we tried both Conjugate Gradients [PR69; She94] and BFGS [NW06] as optimization strategies, which produced similar results and we report the BFGS results as it ran faster.

To parameterize our rigid-body transformation, we perform the optimization on $\mathbb{R}^7$, with a translation $t \in \mathbb{R}^3$ and a quaternion $q \in \mathbb{R}^4$. Quaternions are well studied in the context of optimization for rigid-body transformation [SN01; HK09]. As we use numerical differences in our optimizer, we did not utilize methods the closed-form gradients for quaternions [XXM16]. While many authors prefer the exponential map for optimizing rigid-body transformations [TOM18], our experiments using the rotation vector $v = \theta\hat{v} \in \mathbb{R}^3$ (with rotation angle θ around the unit vector $\hat{v}$) produced nearly identical results in our final registration result.

We performed these experiments on our mesh and point-derived Gaussian Mixture Models, as well as two baselines. We implemented our own point-to-point Iterative Closest Point (ICP) method [Pau92] and used an exiting implementation of Coherent Point Drift (CPD) [MS10] from pyCPD [Kha19]. We adjusted pyCPD to run for 150 iterations to approximately match the run-time of our P2D GMM registration. ICP we ran for up to 50,000 iterations, or until the improvement in mean matching error was below 10^{-9}. For consistency, all methods used in this chapter were implemented in the Python programming language and only used the CPU.

4.6.2 Analysis of Mesh Registration

The results in Figure 4.6 demonstrate that our mesh-derived Gaussian Mixture Model provides improved registration results when using P2D compared to the existing baselines, ICP and CPD. Not only does our method produce better registration on average, but it also demonstrates a better distribution of errors. Specifically, the small difference between the median and mean errors shows that our method is less prone to outliers. On the other hand, the point-based GMM P2D registration results had outliers that dragged the mean towards the worst quartile of results. The randomly initialized point-based GMM had a mean that was about three times that of its median result, suggesting that some experiments produced results in the incorrect local minima.

4.6.3 Other 3D Models

We report results on additional 3D models in Table 4.1. For consistency, we decimated each model to 1000 faces using [GH97] and then repeated our previous experiments exactly (except that the registration results are now the average of 25 runs). The likelihood column reports the per-sample average log-likelihood of ground truth, where larger numbers are better. The translation and rotation errors are reported as a percentage of the error obtained by ICP registration. In all our experiments, the mesh-based GMM always outperformed the point-based one, often significantly.

Model	**Likelihood** (larger is better)		**Translation Error** (% of ICP)		**Rotation Error** (% of ICP)	
	points	mesh	points	mesh	points	mesh
Armadillo	-14.6	-12.2	127	37	161	33
Bunny	7.6	8.2	50	28	41	17
Dragon	6.9	7.6	68	25	40	19
Happy	7.3	8.2	101	27	85	27
Lucy	-21.5	-18.3	95	23	122	35

Table 4.1: Results of repeating the experiments in Figures 4.3 and 4.6 on mulitple models from [Lev+05]. All experiments used k-means initialization and $K = 100$. Details in Sec. 4.6.3.

4.6.4 Visual Odometry

Our proposed method can be also used for improved models of partial view observations, such as those seen in simultaneous localization and mapping (SLAM). In this case, the geometric primitives used represent not the exact surface, as with our mesh experiment above, but instead an uncertainty region for each 3D measurement.

We performed experiments using a GMM distribution-to-distribution (D2D) registration method for visual odometry [TOM18], reproducing an experiment on an RGB-D dataset sequence from the TUM dataset (`freiburg3 long office household`) [Stu+12].

We are able to incorporate structural information into the fitting of the Gaussian Mixture Models by adding depth uncertainty information around each 3D point and applying equation 4.13 during GMM fitting. The results are shown in Figure 4.7.

For our registration experiments, we first subsampled the depth images to 160×120 resolution before performing frame-to-frame registration over the 2510 frame sequence. The ICP method used our aforementioned point-to-point ICP method over 2,500 points randomly sampled from each point cloud (selected to roughly match the run-time performance of our method). The GMM method fit a $K = 100$ GMM to the point cloud using our uncertainty model, and performed registration using a D2D metric [TOM18]. We used the determinant-free method as it was much faster in our re-implementation. Additionally, our implementation used numerical gradients and BFGS [NW06] as the optimizer.

In this case, our primitive model was simple one, a rectangle representing the size of each 3D measurement in the X and Y axes of the camera. This generated a trajectory with absolute translational error RMSE of $0.878m$, a small improvement (2.4%) over the $0.899m$ APE produced from building GMMs without uncertainty primitives. Additionally, we found that D2D registration time was 22% faster when using GMMs built from primitives.

4.7 Conclusion

We have shown how to build Gaussian Mixture Models by incorporating structural information into their Expectation Maximization algorithm. We demonstrate theoretical and empirical equivalence with traditional techniques, along with providing a fast approximation to our proposed method. By using the covariance structure from the triangles of a mesh, we are able to build GMM models more quickly, robustly and to higher quality. Additionally, these models provide an improved result in 3D registration. For a theoretical understanding of how geometric structures, point samples, and integrals interact, our product integral derivation provides a model that is invariant to resampling (such as triangles being merged or split while retaining the same overall 3D structure). We believe that chapter demonstrates that using structural information can lead to methods that are faster, more robust, and more lead to improved performance.

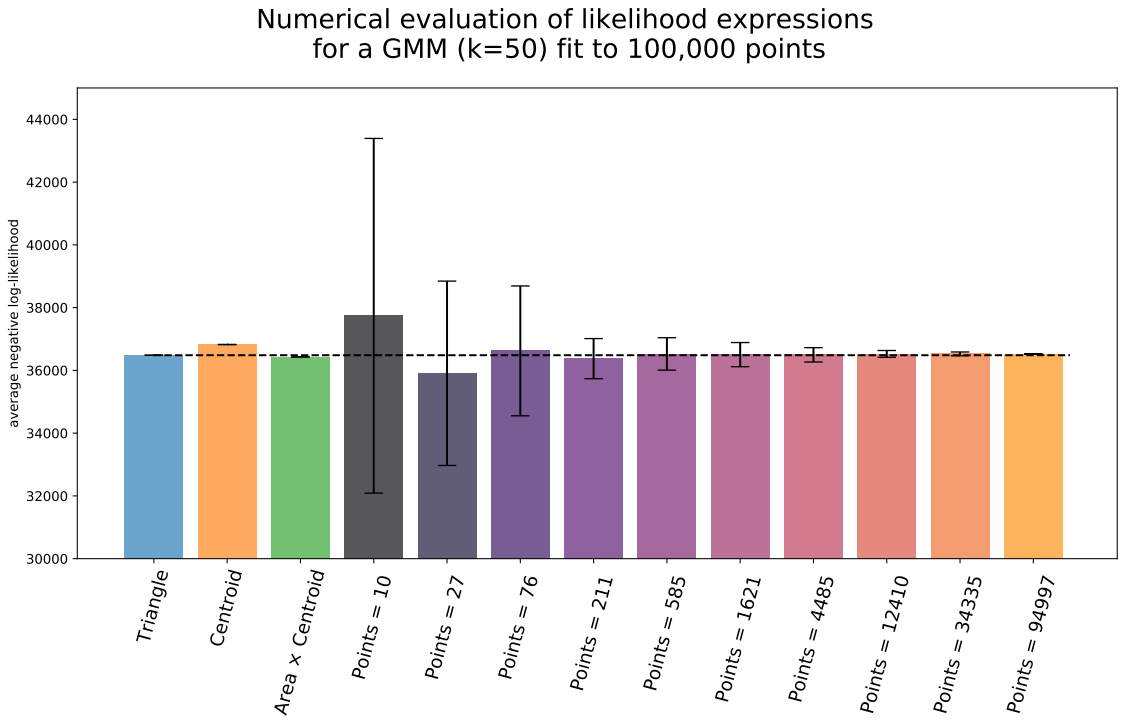


Figure 4.2: **Comparison of fitting metrics**. After fitting a 50 mixture GMM to 100,000 points randomly sampled from the *res4* Stanford bunny, different log-likelihood expressions are compared. *Triangle* refers to the equation 4.8, while *Centroid* refers to using only the triangle centroids, while *Area* $\times$ *Centroid* refers to using our approximation in eq. 4.5. The results are on the *res4* variant of the Stanford bunny, which has 948 faces and 453 vertices. The remaining bars show results using different numbers of points sampled from the mesh surface. The y-axis shows the sum of individual Gaussian component log-likelihoods ($\sum\sum \log(x)$), equivalent to the lower-bound obtained from Jensen's inequality. The horizontal line shows the result of using all the points, our best approximation of the correct answer.

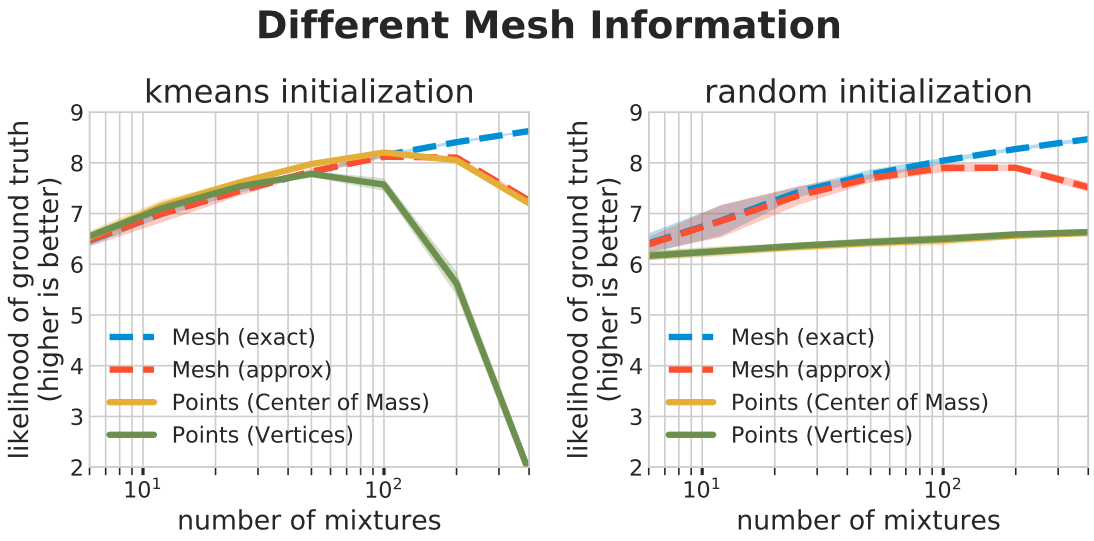


Figure 4.3: **GMMs fit to different data-types of the low-res Stanford Bunny**. The graphs show fitting fidelity of the converged model. The dashed lines use triangle likelihood estimates, while the solid lines use traditional point loss. *Exact* refers to the M-step derived in eq. 4.9 & 4.10, while *Approx* refers to only using eq. 4.5. The results are on the res4 variant of the Stanford bunny, which has 948 faces and 453 vertices. Evaluation is performed by evaluating the likelihood of 50,000 points sampled from the *res4* Stanford bunny.

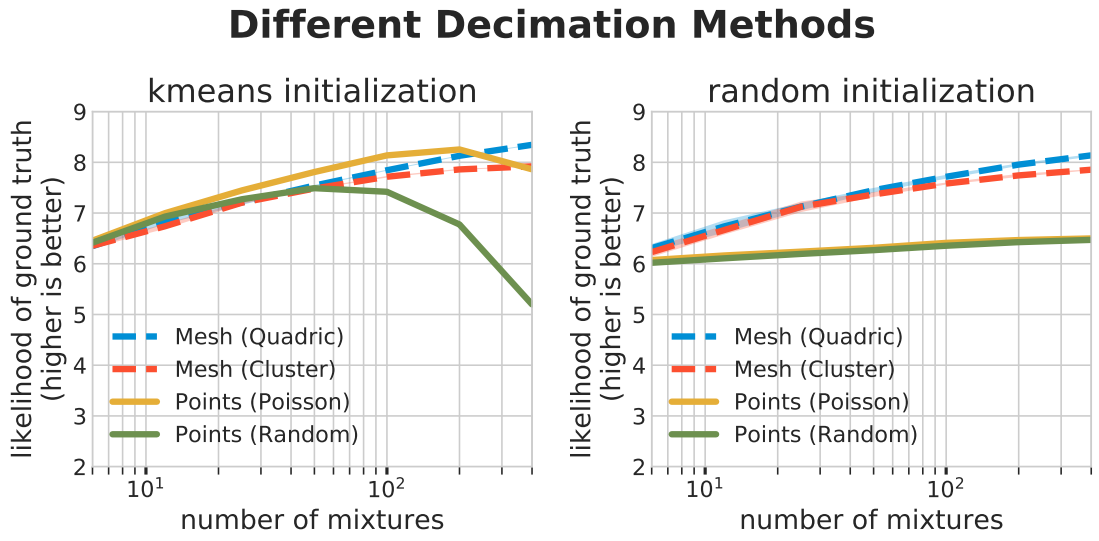
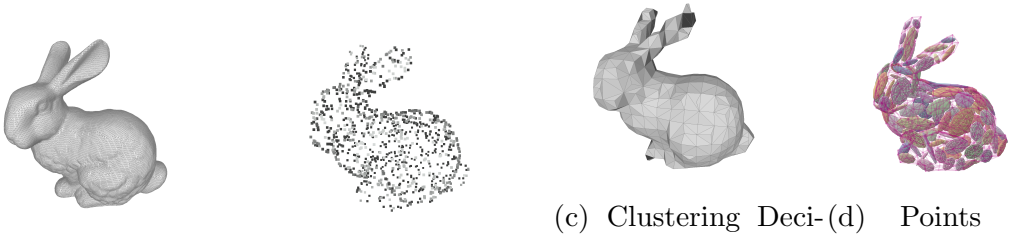


Figure 4.4: **GMMs fit to different input formats of the Stanford Bunny**. The graphs show fitting fidelity of the convered Gaussian Mixture Model. The dashed lines use eq. 4.9,4.10, while the solid lines use traditional point loss. The results are on the Stanford bunny, which has been simplified to ≈ 1000 triangles or points respectively with two different methods each. Random or Poisson disc sampling [CCS12], and with either clustering [RB93] or quadric-error decimation [GH97]. See Fig. 4.5 for visual examples of these formats. Evaluation is performed by evaluating the likelihood of a high-resolution point cloud sampled from the original Stanford bunny.



(a) Stanford Bunny

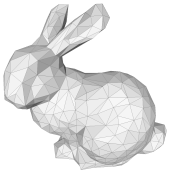
(b) Random Samples

(c) Clustering Decimated [RB93]

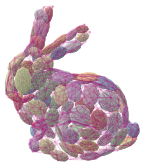
(d) Points GMM ($K = 100$)

(e) Dense Sampled

(f) Poisson Disc Sampled [CCS12]



(g) Quadric Decimated [GH97]



(h) Mesh GMM ($K = 100$)

Figure 4.5: Examples of different input and output representations for the Stanford Bunny

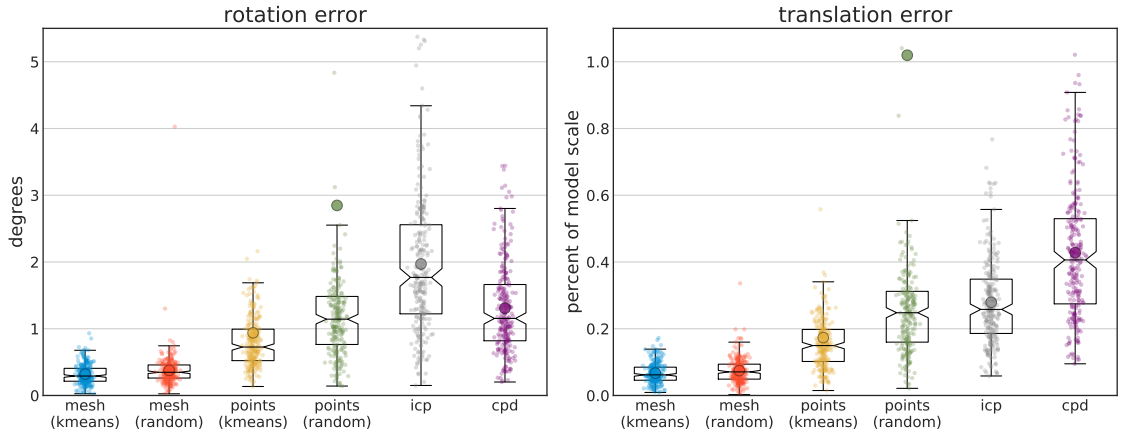


Figure 4.6: Registration results on the Stanford Bunny, following the experimental setup described in in [EKK18]. The experimental conditions for these tests are described in Section 4.6. We plot the results of 250 random rigid deformations. We show the actual data, along with a box and whisker plot showing the median and its confidence interval; the mean is plotted as a larger dot. *mesh* and *points* are the results of maximizing the likelihood of a set of points (N=453) against our fit Gaussian Mixture models (K=100). The mesh result uses our proposed method, fitting a GMM to the mesh triangles, while points shows the results of fitting a GMM to the mesh vertices (N=453). *icp* is our implementation of point-to-point ICP [Pau92], and *cpd* [MS10] is from pyCPD [Kha19]. *Model size* refers to the length of the diagonal of the model's bounding box, and we report our results in percent (so all reported methods have position error, on average, better than 1.1% of the model size). Some methods have outliers that converged to the wrong local minima, and hence have a very large mean relative to their distribution.

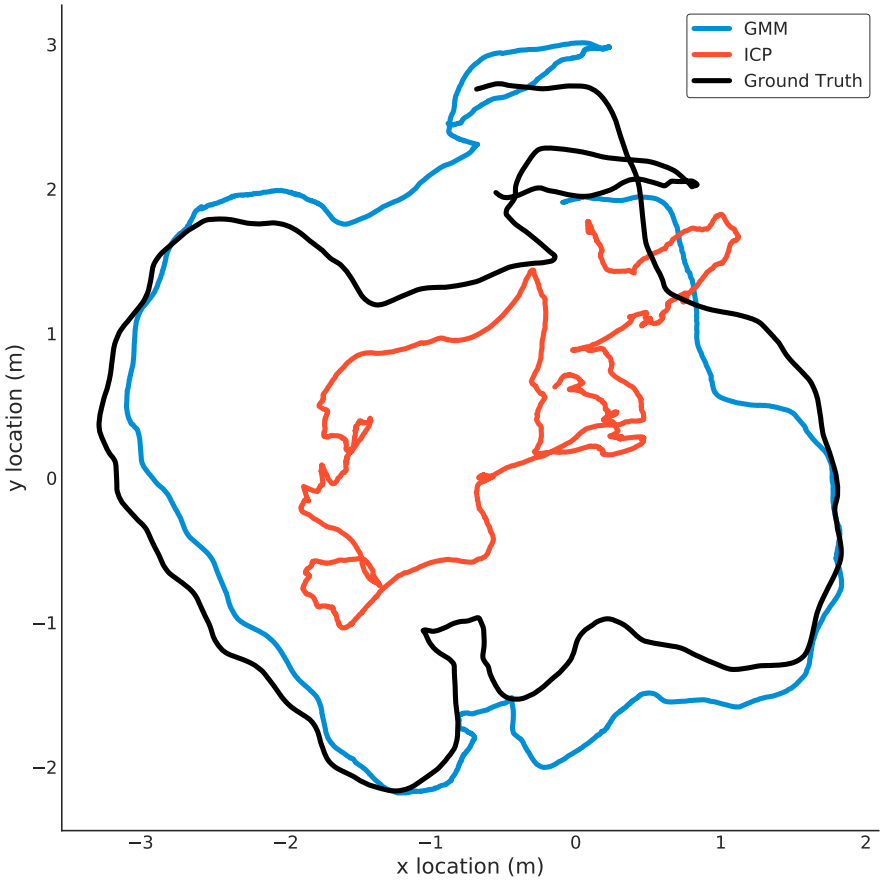


Figure 4.7: Top-down view of trajectories generated using different registration methods for visual odometry. The GMM method uses a per-pixel uncertainty primitive during GMM fitting. The ICP method is pt2pt ICP [Pau92]. Runtime for both methods was similar. For details, see Section 4.6.4. For additional baselines and experiments on this dataset see [TOM18].

Chapter 5

Discovering Multiple Algorithm Configurations

This chapter focuses on leveraging the variance seen during the black box tuning of algorithms across a dataset. We use this variance to find a partition of the dataset where different datapoints may prefer different types of configurations. This lets us explore properties of the dataset, under the view of algorithms trying to perform well across the entire dataset. This chapter is largely similar to our published manuscript on the topic [KH23a].

5.1 Introduction

Autonomous integrated systems often depend on a multitude of algorithms interacting with each other and their external environment. Despite the recent popularity of deep, end-to-end trained models [Ope+19], robotic systems often depend on hand-designed algorithms in several parts of the processing stack. They include motion planning [Col+14], algorithms involved in sensing [Kes+17] and simultaneous location and mapping [EKC16]. As systems become more sophisticated, they often accumulate more methods and with them, more parameters that need to be set and configured by the system designers.

Often the developers of these methods can discover viable configurations by hand but leave many settings open to configuration by eventual users. Intuitively, these

can include settings can control smoothing, performance and run-time. Ideal settings in noise-free environments can vary dramatically from those required in noisy settings. Likewise, different configurations may exist for optimal online and offline performance. Tuning such settings to work well in a deployment environment remains a challenge for many autonomous systems. Without proper tuning, components that are expected to be reliable may unexpectedly fail.

While it is possible to tune settings by hand, it is also possible to use automated methods to find potential configurations. In classic computer science literature, this was done to optimize runtime performance [Ric76] and was known as the algorithm selection (or configuration) problem. Researchers have shown how automated tuning can quickly and robustly improve the performance of even common tools such as compilers [Ans+14]. In an era with many benchmarks [GLU12; MG15; Sch+14] and challenge competitions [Kro+18], algorithm tuning is performed on a validation set made to approximate the performance of the final test set, and ensure the best possible performance for proposed techniques. To demonstrate consistency and generalizability, researchers often report performance under a single configuration.

Considering multiple configurations can greatly expand the applicability of a particular method [Mak+21]. In the case of multi-objective optimization, a Pareto set exists, where progress on any one objective regresses performance on another objective. As such, providing multiple configurations is often done by hardware vendors [Kes+17], and selecting between multiple configurations in robotics is also well studied [HK17b; HK17a; HK18], along with selecting from ensembles of solvers [CAS15] and motion-planners [Tal+16].

In this work, we propose to discover multiple viable configurations while an algorithm configuration is being automatically tuned over a dataset by some black-box optimizer [Han16; LH16]. By noticing correlations between data in response to new configurations, we detect multiple algorithm modes. Working in algorithm configuration space enables generalization across several problem domains, as they do not depend on domain specific features. This allows our method, with fixed settings, to show benefits in multiple application areas (see Section 5.4). We show how multiple modes can be found online (Section 5.3.5) and how they can be used to guard against outliers (Section 5.4.2).

5.2 Related Work

There is a long history of work in algorithm configuration and related areas. Originally studied for performance optimization [Ric76], algorithm configuration has been well studied in the SAT solver community [Ans+15]. These extensions include large evaluations [Han+21], taxonomies of methods [Sch+22] and methods which adapt the algorithm configuration over a series of time-steps [Eim+21; YDB21; Adr+22], assuming the algorithms used have different temporal properties.

Since robotics applications and datasets are reasonably expensive operations (compared to purely synthetic tasks), our work is related to work on extremely few function evaluations, typically on the order of 100 [Ans+21]. Typically this is studied in the Machine Learning community as hyperparameter search, finding configurations for ideal neural network configuration and training [Kot+19; LH16].

Our work is most closely related to portfolio-based algorithm configuration literature [Ley+03; GM04; LNS09]. These methods often design a different configuration for each instance of the problem in their dataset (e.g. each data example) [MS09; XHL10; Kad+10]. Similar to our work, they cluster methods into groups. However, their clusters are derived from domain-specific feature spaces, while we use the response of the instances to new configurations. Our use of domain-specific features is limited to test time deployment, using supervised training to target our obtained algorithm configuration clusters.

In the Machine Learning community, the problem of coreset discovery [MBL19] is related to our approach. Coreset discovery finds representative examples from a dataset to focus training time on a subset of the data. Most related, methods exist for online discovery of such sets [Yoo+22]. Of note, our contribution is orthogonal to coreset research, as our method could benefit from coreset methods, which would serve to give us a smaller subset of data to evaluate at each iteration. Specifically, coreset methods for clustering [BLK17] could enable more function evaluation steps in a fixed budget of time and give better resulting minima.

In computer vision, some recent work has explored estimating algorithm configurations for classic algorithms on a local level, operating on patches within an image [Wro21].

5.3 Method

Our approach consists of evaluating algorithm configurations from a black box optimizer (Section 5.3.2) across a dataset of examples for the given algorithm (Algorithm 1). Building upon this baseline, we propose three methods of partitioning the data during optimization: Post hoc (Algorithm 2), Staged (Algorithm 3), and Online (Algorithm 4).

For our experiments, we always use two partitions, even when there are more known modes (Section 5.4.1). This avoids exploring the area of instance-specific algorithm configuration and minimizes our risk of overfitting to the dataset and overstating our performance. We typically report results between the initialization (the known defaults for an algorithm) and the oracle. Our oracle is defined as awarding the best known configuration for each individual datum across all optimization runs.

5.3.1 Partitioning

The optimal partition for a given number of partitions K, with M algorithm configurations over N datapoints can be formulated via 0-1 Integer Linear Programming where $c_{i,j}$ corresponds to the quality of datum i with configuration j.

$$\begin{aligned} \min \quad & c_{i,j}\, x_{i,j} \\ \text{subject to} \quad & \\ & x_{i,j} \in \{0,1\} \\ & \sum_{j=1}^{M} x_{i,j} = 1 \\ & \sum_{j=1}^{M} \mathbb{1}\left[(\sum_{i=1}^{N} x_{i,j}) > 0 \right] \leq K \end{aligned}$$

One can also exhaustively evaluate all $\binom{M}{K}$ partitions. Our experiments do exhaustive evaluation for $K = 2$ (as with all results in this chapter) and use the optimization formulation with larger numbers of partitions. We solve the optimization

problem with a recent solver [HH18] and implement the indicator variables using the BigM modeling trick.

As an alternative, we evaluate using a clustering method such as k-Means [Scu10b] on a normalized matrix $\tilde{X}$, where each row has zero mean and unit variance. Cluster centers are in the space of the history of evaluated configurations and each row is a datum's response to the history of evaluated configurations. Clustering approaches treat the algorithm configuration history as a feature and group results which behave similarly, but may not be optimally partitioned.

5.3.2 Black Box Optimizer

Our method is generic to the choice of black box optimizer, also known as gradient-free optimization. For example, one could use random search, which is known to be a strong baseline in higher dimensional optimization [Gol+17]. On the other extreme, if one has expensive function evaluations, one could fit a surrogate function to the data and optimize its expected minima, as is done in Bayesian Optimization [Fra18]. Effective black box optimizers in practice often combine a plethora of optimizers [Ans+14; RT18].

We use CMA-ES, an evolutionary method with a multivariate Gaussian model [Han16]. CMA-ES is known for algorithm configuration search in robotics [Zha+17] and is widely used in other algorithm configuration comparisons [Eim+21]. CMA-ES is convenient in only requiring an initial configuration and a σ in parameter space. When tuning existing algorithms, reasonable prior configurations are often available. Approaches which focus on modeling a bounded volume [Fra18; RT18]) can be wasteful. All of our parameter search is for non-negative parameters, so we transform our search space with $\log(x)$ to perform unconstrained optimization, making our search operate on order-of-magnitude scale for all parameters.

5.3.3 Post hoc Partitioning

The post hoc method is simple and straightforward: perform black box on the dataset as a whole, noting each datum's response to each configuration. Afterwards, partition the data following Section 5.3.1. This approach allows clearest evaluation against the non-partitioned CMA-ES baseline, which it outperforms in all of our experiments.

Algorithm 2 outlines the *Post hoc* method in detail. As a method with no change in exploration, it can be be run on existing single mode optimization or simply using coherently evaluated random configurations [Gol+17], as in Section 5.4.6.

5.3.4 Staged Partitioning

In staged partitioning, we spend half of the function evaluations exploring the space to find adequate minimia, and we spend half of the function evaluations exploiting the discovered partitions in isolation. While the scale of a particular problem may suggest different balance of exploration and exploitation stages, we use a ratio of $\frac{1}{2}$ for our experiments. Algorithm 3 performs partitioning in the middle of optimization and tunes the results for each partition. This enables more explicit exploitation of the partitions, at the expense of less exploration time to find good partitions.

5.3.5 Online Partitioning

To balance exploration and exploitation in an online fashion, one could use a multi-armed bandit. Algorithm 4 dynamically assigns data points to partitions during the course of optimization. Since CMA-ES only evaluates relative order, we can readily switch data assigned to each partition during the course of evaluation. For the online method, we setup a multi-armed bandit (MAB) for each datum. Since our function evaluations are unbounded, we use classic Thompson Sampling [Tho33; Rus+17] with a Gaussian distribution. We perform one CMA-ES step to sample the space, and use that to initialize the distributions for each arm of the bandits identically. Each iteration, we sample a partition assignment for each bandit. That datum then evaluates the configuration given by that optimizer and records its result for that arm. The optimizers record the mean cost of the data assigned to them that iteration.

This approach allows us to simultaneously perform multiple optimizations and partition assignments on the fly.

5.4 Experimental Results

We evaluate our approach on several application domains. We start with a synthetic function whose structure and modes are known and is quick to evaluate. This

Algorithm 1 Optimize algorithm configuration over a dataset

Require: x_0 Initial Configuration
Require: $f_{1...N}(x)$ dataset queries for algorithm
Require: M Maximum number of function evaluations
procedure OPTIMIZE($x_0, f_{1...N}(x), M$)
  for $i \leftarrow 1$ to N **do**
    $x_i \leftarrow$ OPTIMIZER CANDIDATE()
    for $j \leftarrow 1$ to N **do** ▷ Evaluate all data
      $X_{i,j} \leftarrow f_j(x_i)$
    end for
    $g_i \leftarrow$ MEAN(X_i)
    OPTIMIZER TELL(g_i) ▷ Report average
  end for
  $Y_i \leftarrow$ MEAN($X_{i,j}$) ▷ Per configuration scores
  $x \leftarrow x_{\text{argmin}(Y_i)}$ ▷ Best configuration
end procedure

Algorithm 2 Finding modes with post hoc partitioning

Require: K Number of partitions
Ensure: $x_{1..K}$ Per partition configurations
Ensure: $c_{1..N}$ Per datum partition assignments
procedure POSTHOC($x_0, f_{1...N}(x), M, K$)
  OPTIMIZE($x_0, f_{1...N}(x), M$)
  $c_j \leftarrow$ PARTITION(X^T, K) ▷ Get partitions for data
  $Y_k \leftarrow$ MEAN($X_{i,(c_j=k)}$) ▷ Per partition scores
  $x_k \leftarrow x_{\text{argmin}(Y_k)}$ ▷ Configuration for partitions
end procedure

Algorithm 3 Finding modes with staged partitioning

procedure STAGED($x_0, f_{1...N}, M, K$)
  POST HOC($x_0, f_{1...N}(x), \frac{M}{2}, K$)
  for $k \leftarrow 1$ to K **do** ▷ Separate optimization
    OPTIMIZE($x_0, f_{c_j=k}(x), \frac{M}{2}$)
  end for
end procedure

Algorithm 4 Finding modes with online partitioning

```
1: procedure ONLINE(x_0, f_{1...N}, M, K)
2:     B_k ← BANDIT(N)                          ▷ K arms for each datum
3:     OPT_k ← OPTIMIZER()                      ▷ K optimizers
4:     for i ← 1 to M do
5:         for j ← 1 to N do
6:             b_j ← BANDIT PULL(B_j)           ▷ Sample bandit
7:         end for
8:         for m ← 1 to K do                    ▷ Separate Evaluation
9:             x_{i,k} ← OPT_k CANDIDATE()
10:            y_{i,k} ← MEAN(f_{b_j=k}(x_{i,k}))
11:            OPT_k TELL(y_{i,k})
12:        end for
13:    end for
14:    c_j ← BEST ARM(B_j)
15:    y_k ← BEST CONFIG(OPT_k)
16: end procedure
```

enables us to characterize our different methods of finding partitions. We then proceed to show successful benefits to robotics methods like stereoscopic depth generation [Kes+17], differentiable rendering [KH22], motion planning [Lav98], and visual odometry [EKC16].

5.4.1 Synthetic Function

A synthetic function allows us to characterize our methods across arbitrary many dimensions and modes. Our synthetic function has K modes, each the sum of N hard-to-optimize functions, leading to a simulation of KN data points. We use four hard-to-optimize functons: Ackley, Griewank, Rastrigin, Zakharov (for details of these functions and their visualizations see [Lv+18]), and rescale them to have a minima of value zero, a random rotation, and to have a maximum value of around one near the minima. This paradigm and these functions can be generated in arbitrary many dimensions, allowing us to understand how these partitioning methods scale as algorithm hyper-parameters scale from two to forty dimensions.

For the synthetic function optimization, the staged method works best across most dimensions and number of function evaluations. Close behind, especially with fewer

evaluations, is the post hoc method. In contrast, our online bandit method is typically only slightly better than the single mode baseline. Of note is that all methods begin to perform better with hundreds of function evaluations, suggesting that the improved performance of the partitioning may come from improved efficiency in low numbers of evaluations, and not the multi-modal nature of the synthetic function.

5.4.2 Dense Stereo Matching

Robotics applications often use stereoscopic depth sensors. Here we optimize the performance a classic Dense Stereo Matching method, namely Semi-Global Block Matching (SGBM) [Hir08] as implemented by OpenCV [Bra00]. We obtain 47 image pairs by combining the Middlebury 2014 and 2021 Stereo datasets [Sch+14]. We split the data into 23 training examples and 24 test examples, shown in Section 5.4.2. The algorithm settings control the regularization of the SGBM algorithm, the post-processing filters used to cleanup the data, and the block size used for initial matching. Results of the four methods on the training set are shown in Fig. 5.5.

In deploying the discovered configurations to new data, we show the efficacy of a simple supervised classifier. The classifier used is k-nearest neighbors with a $k = 1$, returning the partition index to be used. We use a pre-trained neural network's top level feature space as the feature space. Specifically we use SqueezeNet 1.1 [Ian+16] pre-trained on ImageNet in PyTorch [Pas+19] and its 512 dimensional space for images.

The test set performance is improved with partitioning, as shown in Fig. 5.6c with quantitative estimates and Fig. 5.6b with two qualitative examples from the test set.

We find that the optimal hyperparameter configuration typically focuses on regularization and filtering. The first configuration usually has less regularization, but a more aggressive filter to discard bad matches, while the second configuration has more regularization and less aggressive filters to discard bad data.

5.4.3 Differentiable Rendering

We optimize the parameters of a recent differentiable renderer [KH22]. Our dataset includes 20 sequences from the KITTI odometry dataset [GLU12] and 20 synthetic shapes. The KITTI sequences use k-Means to build a quick model of 10 consecutive

LIDAR frames, from the center of the scene looking out. In contrast, the synthetic sequences all have the object densely in front camera. The differentiable render has four hyperparameters, two controlling the sharpness of the silhouettes, one controlling surface smoothness and one controlling how opaque objects are. We optimize all four for depth and silhouette accuracy, similar to the original paper.

In rendering, the optimizer is unable to find a better single mode configuration than the initialization. However, all proposed methods show a statistically significant improvement over the baseline, with the staged method performing the best. In addition, we report the ability of the methods to properly partition the disparate datasets in Fig. 5.7a.

5.4.4 Motion Planning

We evaluate a popular motion planning method, Informed RRT* [GSB14] in the Sampling-Based Motion Planner Testing Environment [Lai21]. We setup three start-goal pairs for three testing environments. We use the geometric mean of runtime (as estimated by the number of expanded notes) and performance (as estimated by the quality of the first found solution). This balance of runtime and quality is essential to obtaining an interested configuration. Results are shown in Fig. 5.8, with only the post hoc method outperforming the baseline. Other methods perform poorly, and we suspect the problem is insufficient samples and lack of exploration in the online and staged methods; especially as RRT*-based planners are stochastic, making evaluations noisy.

We find that the optimal partition finds different parameters focused on the goal sampling frequency (0.2 and 0.3; single mode 0.26) and the rewiring radius (1500 vs 7500; single mode 6000). Of our three start/goal pairs in each of three different environments, optimal partitioning typically grouped one environment together.

We also performed some experiments with RRdT* [LRF19], which we report briefly report. Often, one partition would focus on a configuration that frequently spawned new trees, while the other focused on expanding existing trees.

As it is unclear how to parameterize motion planning goals and environments for supervised classification, we were unable to do experiments on a hold-out test set.

5.4.5 Visual Odometry

We perform experiments on a subset of the TUM VI Visual-Inertial Dataset [Sch+18] using DM-VIO [SC22]. DM-VIO has many parameters but we focus on five (points, immature points, min frames, max frames, max optimization steps). TUM VI has 5 environments, and we select the third sequence from each environment as our dataset.

We prioritize a geometric mean of runtime (frame time) and quality (best-aligned absolute pose error [Gru17]), while penalizing trajectories which do not complete successfully. Results are shown in Fig. 5.9. Reliably, the algorithm partitions a separate configuration for all but the *slide3* sequence, which includes fast motion through a closed pipe. The slides partition is the best single mode, while the alternative partition uses fewer points (100 vs 350) and fewer frames (2-4 vs 3-5) as it does not need to handle the difficult high-velocity, highly occluded sequence.

As our VO partitions depend on properties of the sequence, we were unable to construct a reasonable test set based on the first frame. Instead, our multiple configurations may be used in on-the-fly configuration selection [HK17a],

5.4.6 Commercial Depth Sensor

Lastly, we demonstrate our partitioning method on a Intel RealSense D435 [Kes+17] and its 35 parameters for estimating depth. We generate a set of 500 randomly configurations. We evaluate all configurations on 10 scenes, for which we have collected their pseudo ground truths using a moving laser pattern [Kes+17]. We partition the configurations using the post hoc method. Results for four scenes are shown in Fig. 5.10.

The optimal $K = 2$ partition included the best single mode configuration as well, allowing us to show it and the alternative configuration. The single mode configuration produced small holes, but dense results outdoors. While the alternative configuration produced smoother, denser walls in indoor environments in exchange for more artifacts outdoors.

5.5 Discussion

Many algorithms in robotics operate in environments with multiple modes. These natural partitions are easy to understand, and can be discovered naturally by analyzing how different datums respond to different algorithm configurations. The modes were found because they affected algorithm response, not because they happened to be grouped together in some domain-specific feature space.

All the proposed methods for partitioning show some efficacy. Across the board, the post hoc method works well. This is likely due our extremely small number of evaluations for algorithm configuration [Ans+21], leading to more benefits for exploration. The online method typically performs poorly in this setting and it is possible that more sophisticated bandit algorithms [MG17b] could perform better.

Our experiments focused on two partitions for all methods. Even when problems had more modes by construction, two partitions were able to clearly improve performance.

5.6 Conclusion

Automatically finding modes during the course of algorithm configuration is a viable way to improve algorithm performance in several different areas of robotics. More study is needed to understand what typical algorithm modes exist and how such modal configurations might be used long-term deployed autonomous systems.

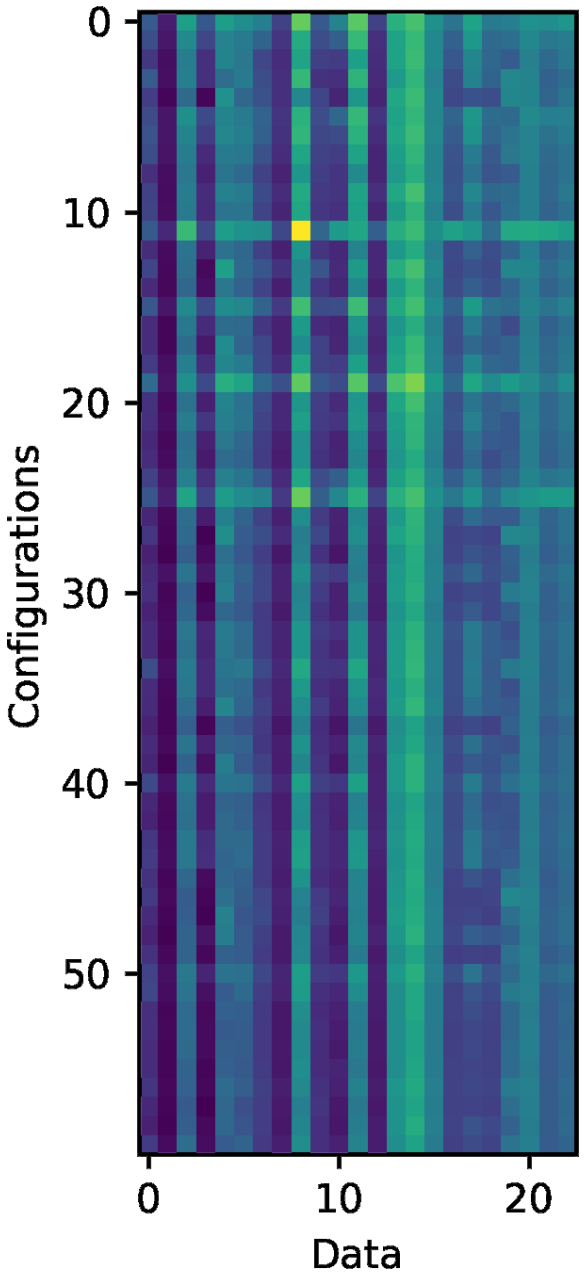


Figure 5.1: **Typical Algorithm Optimization** can be seen as factorized and visualized as different configuration results for each individual datum. This chapter focuses on leveraging this variance in quality to find different modes in the dataset, potentially corresponding to different points on the Pareto front.

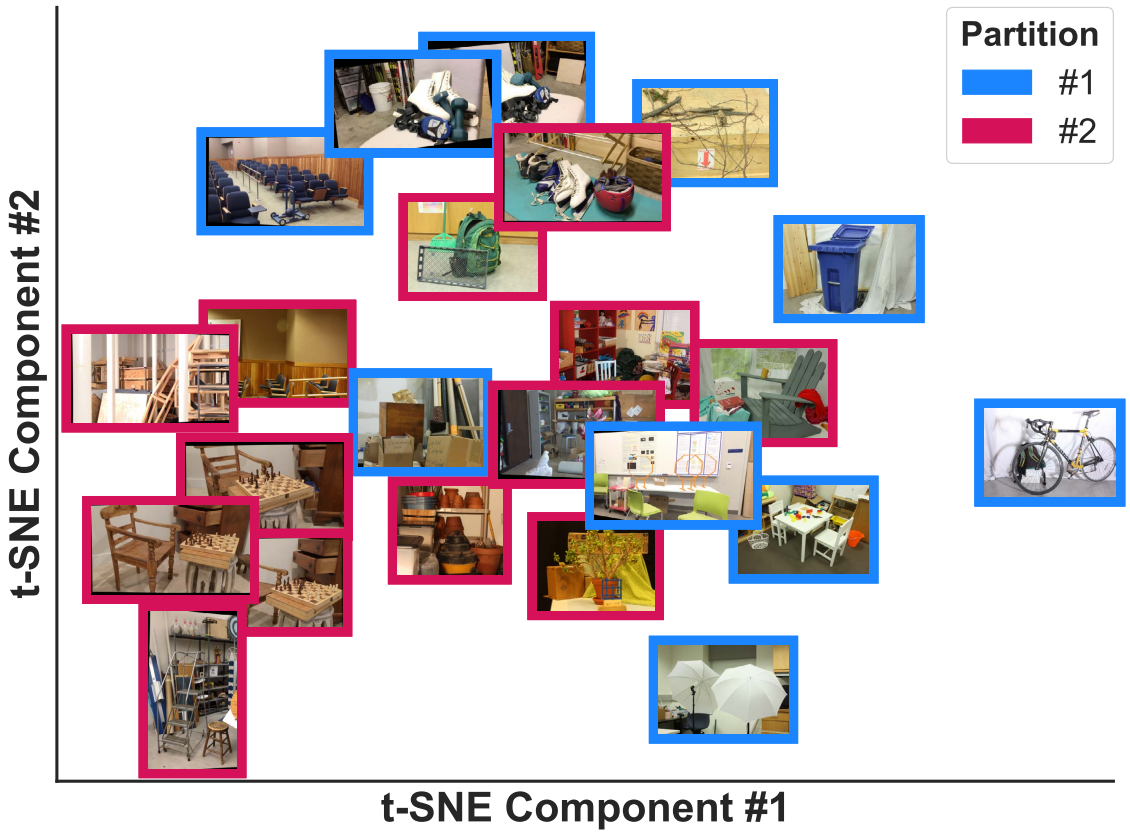


Figure 5.2: **Partitioned Configurations.** Instead of finding a single algorithm configuration for an entire dataset, we partition the dataset during the process of optimization and find a different configuration for each partition.

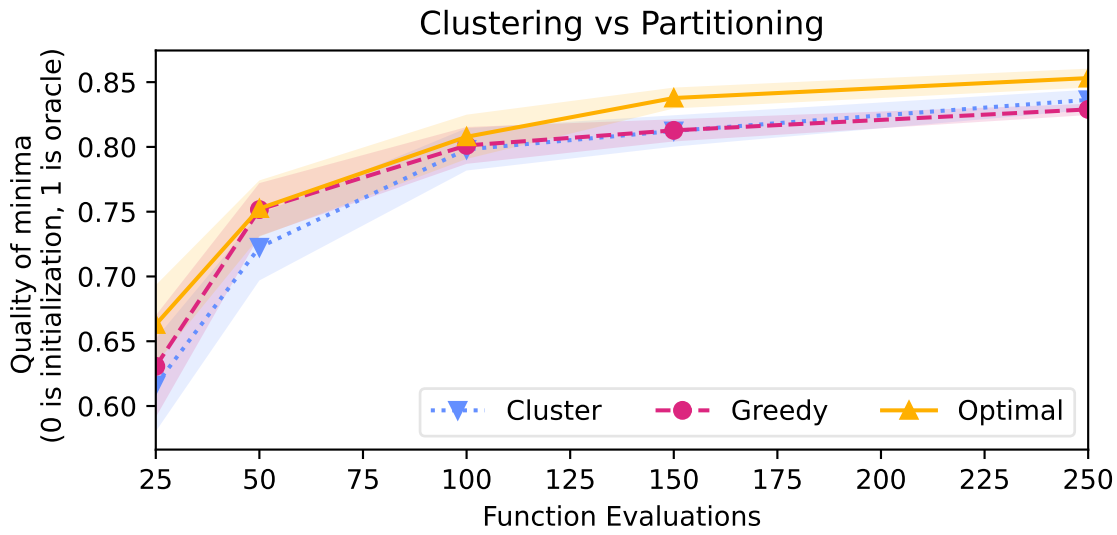


Figure 5.3: **Greedy and Optimal Partitioning vs K-Means Clustering** on stereoscopic depth optimization. Greedy maximizes the marginal value of new configurations.

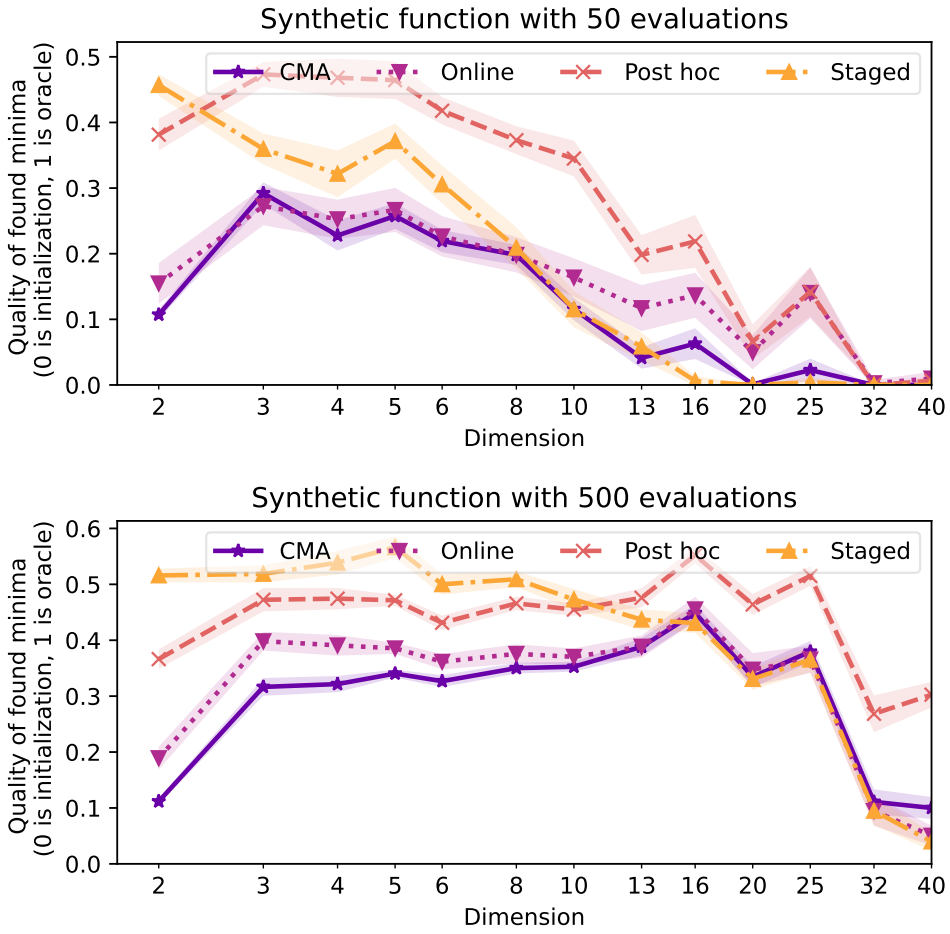


Figure 5.4: **Synthetic Function Partitioning** (Section 5.4.1). Graphs shows the quality of the best found minima for all methods, between the initial configuration and an oracle. Shaded regions indicate standard error of the mean.

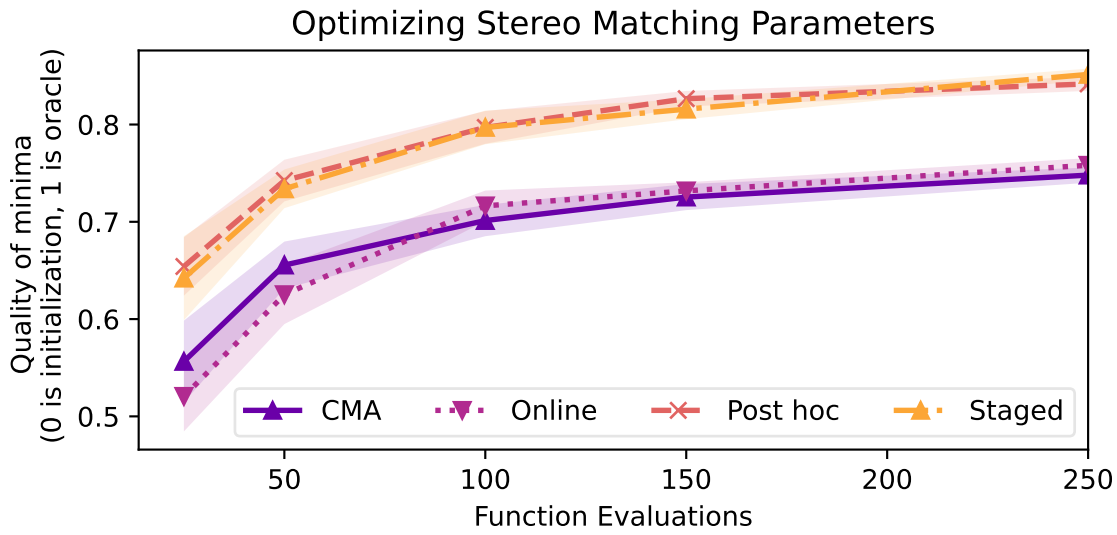
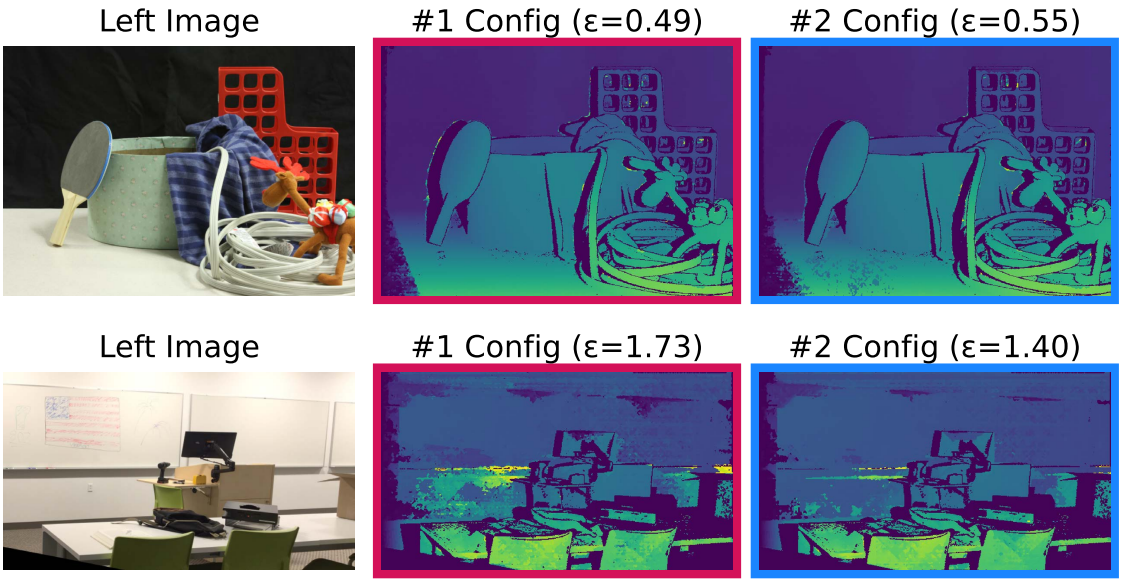
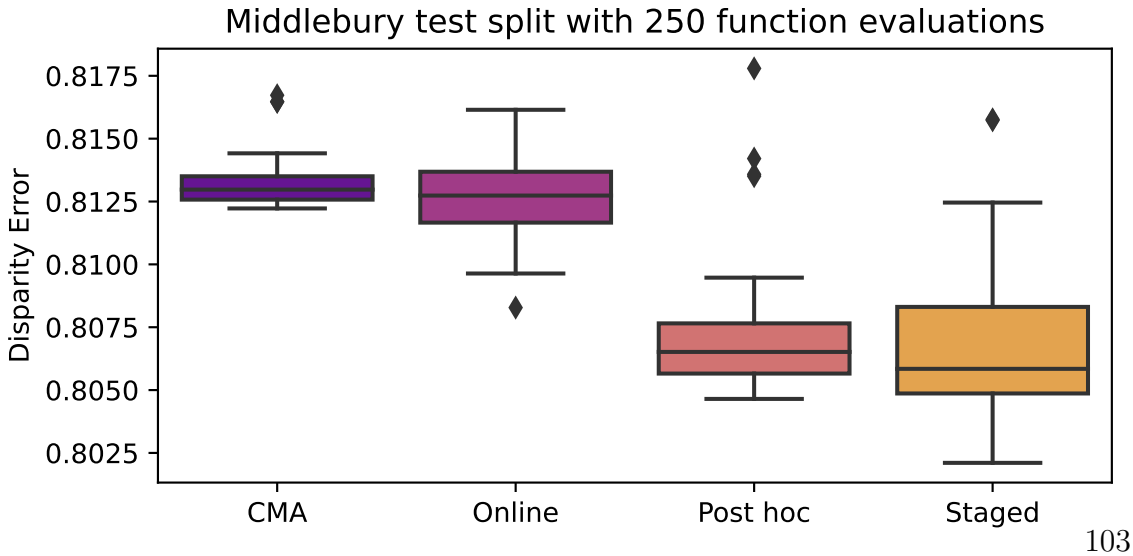


Figure 5.5: **Dense Stereo Matching Partitioning** quality on the training set. The posthoc and staged methods perform well while the online method is indistinguishable from CMA-ES.

(a) **Middlebury Stereo Data** with training data shown as partitioned by our staged method after 250 evaluations. The test set is shown with the predicted classification colors for each image.

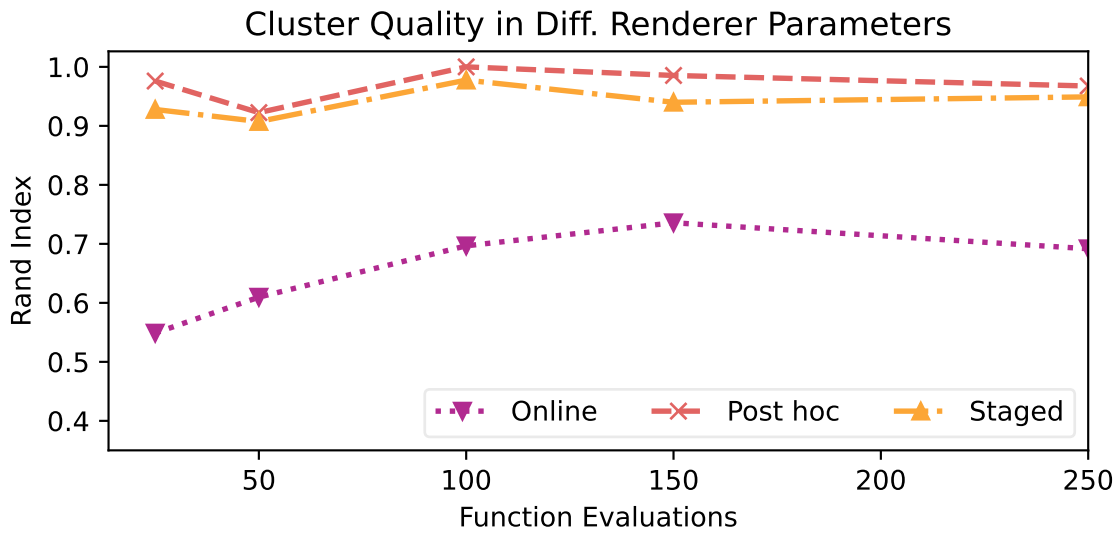


(b) **Qualitative examples of test set results** with our classifier correctly assigning the better configuration to each example. Disparity error shown in parenthesis for each configuration and example.

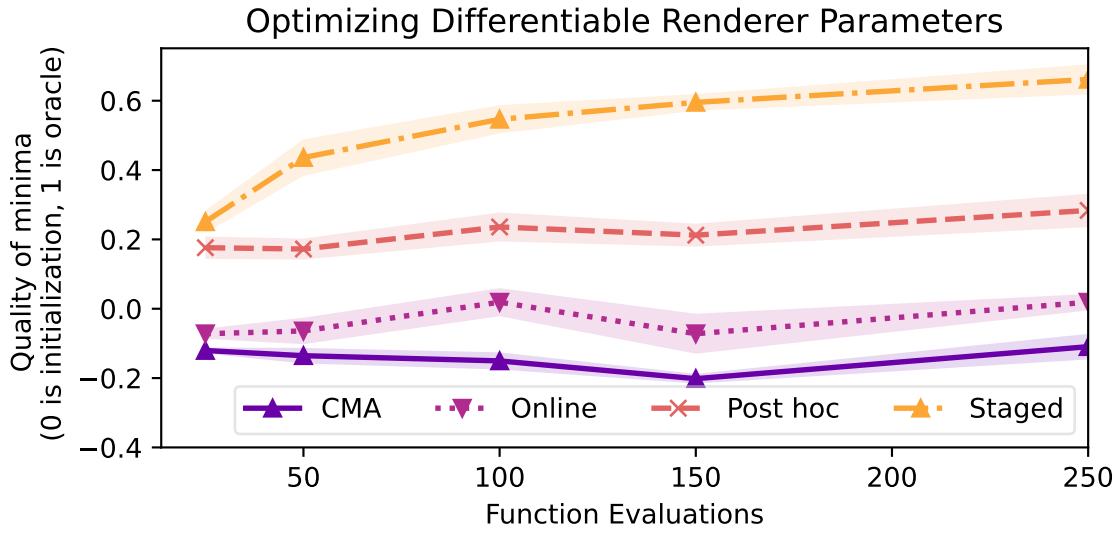


(c) **Test set performance** based on the configurations preferred by our supervised classifier described in Section 5.4.2.

Figure 5.6: **Dense Stereo Matching Test Set Performance**.



(a) **Partitioning Accuracy** in splitting the KITTI data from synthetic data in Section 5.4.3. All methods show some success.



(b) **Differentiable Renderer Partitioning** hyper-parameter optimization for depth and silhouette fidelity. Section 5.4.3. In this case, reasonable defaults restrict the progress of the baseline while the exploitation-focused staged optimization performs best.

Figure 5.7: **Differentiable Renderer Experiments**

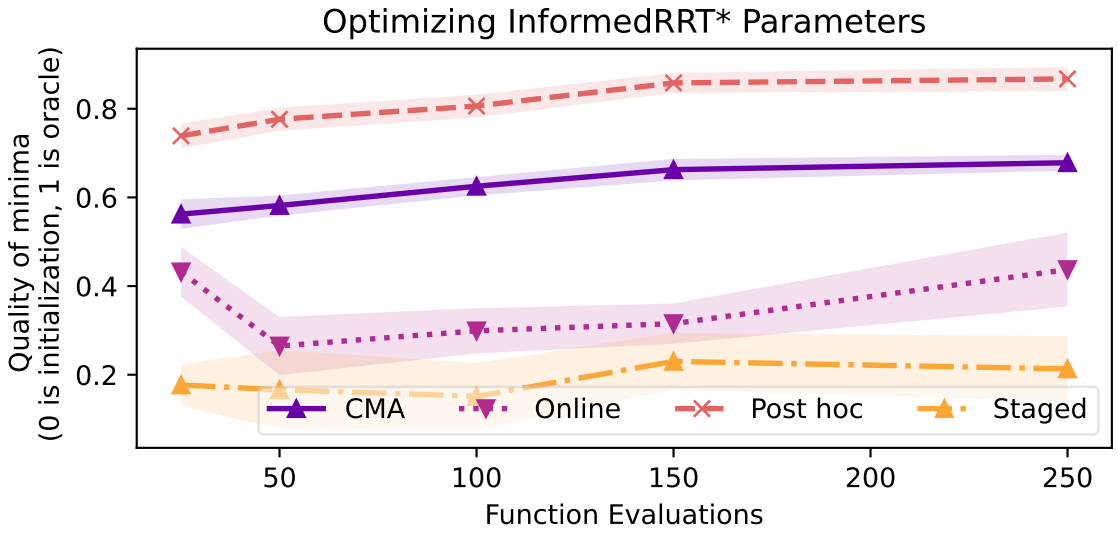
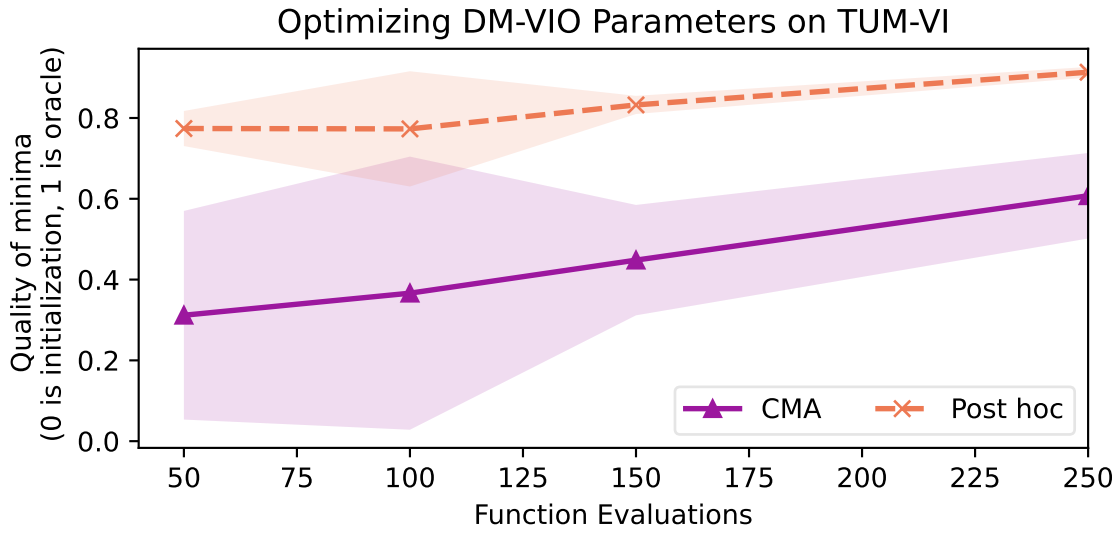


Figure 5.8: **Motion Planner Partitioning** on a set of environments and planar planning tasks. Section 5.4.4 for details.

(a) Partitioned TUM-VI Visual Odometry Dataset



(b) Post-hoc clustering improves DM-VIO performance.

Figure 5.9: **Visual-Inertial Odometry Experiments**

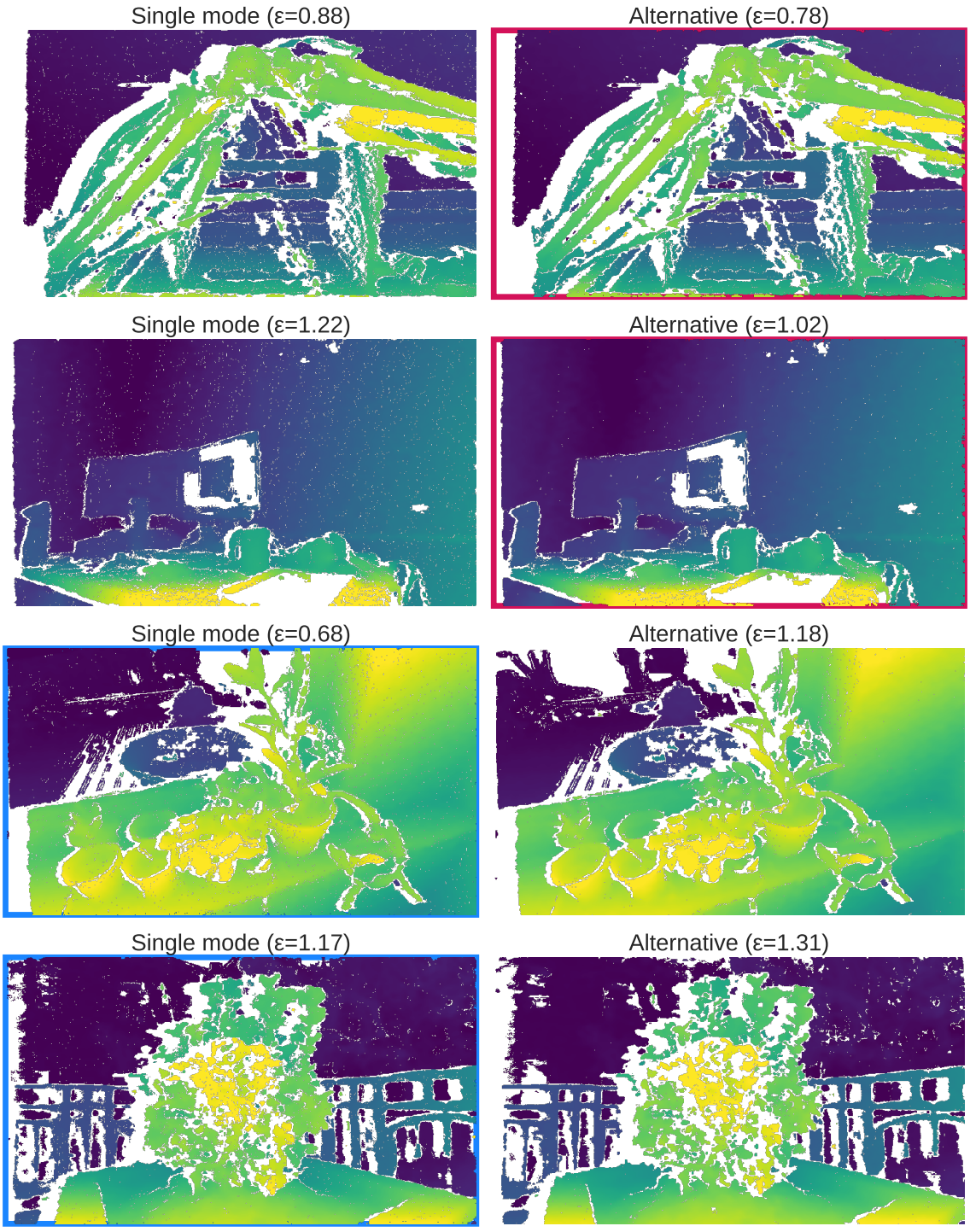


Figure 5.10: **Intel RealSense D435 Partitioning** with 500 randomly generated configurations on 10 scenes. Two are shown, with both configurations (and its error). Section 5.4.6.

Chapter 6

Optimizing From Pairwise User Preferences

This chapter focuses on a method for black box tuning of classic algorithms based solely on user preferences. This is largely similar to parts of published manuscript [Kes+23], which was done in collaboration with others. The publication contains additional experiments and results on social navigation which may be interested to readers, but the content in this chapter focuses just on its contributions to this thesis.

This approach was developed to efficiently tune the 35 parameters in the Intel RealSense depth camera ASIC to achieve better performance. Tuning stereoscopic depth cameras based on ground truth can be difficult, as good edge performance is hard to formalize, since half of all horizontal edges for a stereo depth camera can only be annotated as edges between missing data and foreground objects (due to occlusion regions).

Instead, in this chapter, we show how a modern variant of Random Optimization [Mát65] called CMA-ES [Han16] can naturally tune algorithms based on user preferences with the appropriate interface, as shown in Fig. 6.1. This draws on similar inspiration as the recent approaches for Reinforcement Learning from Human Feedback [Chr+17; Mac+17], widely used in Large Language Models (LLMs) [Tou+23].

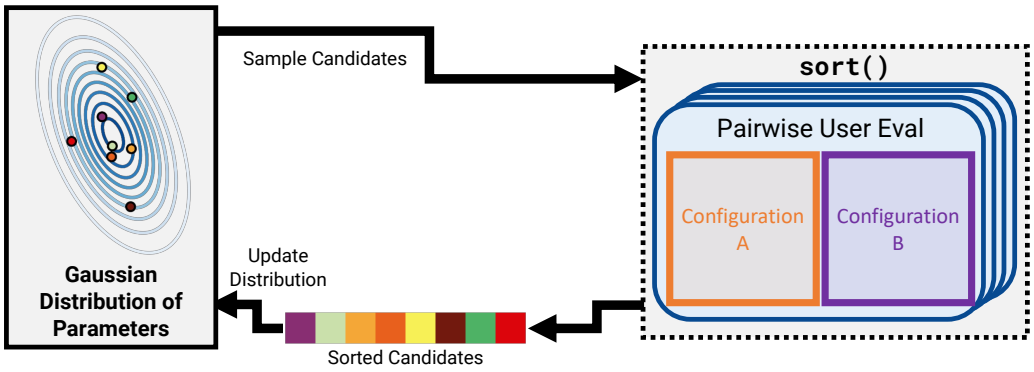


Figure 6.1: **Overview of the proposed approach** for tuning algorithms based on user preferences.

6.1 Method

Our method is designed for finding $\text{argmin}_x f(x)$, $x \in \mathbb{R}^d$, where users select either $f(x_i) \leq f(x_j)$ or $f(x_i) \geq f(x_j)$ and there is no access to $\nabla f(x)$. This represents a d dimensional optimization over a series of configurations $x_{1...N}$: $f(x)$ captures user preferences, and x typically represents the configuration parameters for an algorithm. We call our method SortCMA, since it builds on top of two well-studied techniques: Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Han16] and sorting methods [MG17a].

CMA-ES is a well-known, successful black-box optimization method [Han+21], especially effective when few function evaluations are available [Ans+21]. CMA-ES has been shown to be robust and efficient enough to optimize robotic systems with humans in the loop where the optimization criteria is a continuous metric [Zha+17]. Pairwise sorting is a classic $O(n \log n)$ algorithm with many implementations. For our results, we use Timsort. Extensions to other sorting techniques, such as radix sort, approach direct access to $f(x)$, and our method would be identical to traditional CMA-ES.

SortCMA is a modified interface to CMA-ES that replaces direct evaluation of $f(x)$ with calling a user-querying `sort` function. The `sort`'s `key` function is a user interface that returns the user's preferred selection of $f(x_i) \leq f(x_j)$ or $f(x_i) \geq f(x_j)$. After g generations, when the user is satisfied with the quality, they terminate the optimization and perform $g - 1$ pairwise comparisons to select the final, optimal result.

SortCMA and CMA-ES [Han16] are initialized with a mean $\mathbf{m}^0 \in \mathbb{R}^d$, step size σ^0 and positive semi-definite covariance matrix $\mathbf{C}^0 \in \mathbb{R}^{d \times d}$ (typically $\mathbf{I}$). For each generation of the optimization, a set of $\lambda \geq 2$ offspring are sampled:

$$\mathbf{x}_k^{i+1} \sim \mathbf{m}^i + \sigma^i \mathcal{N}\left(\mathbf{0}, \mathbf{C}^i\right), \quad k = 1...\lambda. \tag{6.1}$$

CMA-ES evaluates all $f(x_k^{i+1}), k = 1...\lambda$, and performs a sorting to assign weights $w_1 \geq w_2 \geq ... \geq w_\lambda$, where better $f(x_k)$ are assigned larger weights. Weights are defined as

$$w_k = \log\left(\frac{\lambda + 1}{2}\right) - \log(k), \quad k = 1...\lambda \tag{6.2}$$

After assigning larger weights to better $f(x_k)$, either via evaluation and sorting (CMA-ES) or direct sorting based on pairwise user evaluation (SortCMA), $\mathbf{m}^{i+1}$, σ^{i+1}, and $\mathbf{C}^{i+1}$ are generated using established CMA-ES update rules [Han16]. The step size (σ) is updated to control the overall exploration rate, and the covariance matrix is adapted to guide the search along favorable vectors in parameter space.

We use *pycma* [Han+23], transform any strictly positive parameters with $f(x) = \log(x)$ for unbounded optimization, and use an initial $\sigma = 0.2$.

6.2 Tuning a Stereoscopic Depth Sensor

Many robots use the widely available Intel RealSense sensors [Kes+17]. The D435 is a stereoscopic depth camera that uses an ASIC-implemented depth algorithm [Hir08], along with a laser [Nis84] for projecting additional texture.

While it is possible to tune stereo algorithms with existing datasets [Sch+14] or on existing camera hardware with pseudo-ground truth [KH23a], these approaches can often be lacking. In particular, real scenes will often contain pixels which are not annotated due to surface properties (specularity, transparency, etc.) or exist

around edges and stereo occluded regions. Additionally, practical usage makes this a multi-objective optimization, balancing depth accuracy, fill rate and outliers (which often appear in unannotated regions).

Instead, we show how a rich scene can be optimized with SortCMA based on, and for, user preferences. This allows optimization without designing a multi-objective loss.

6.2.1 Sensor Setup

The RealSense depth algorithm has 35 parameters that control the properties of the algorithm implemented in hardware. The parameters control regularization of the matching algorithm and checks that ensure invalid results are discarded. By modifying these parameters, users can greatly impact the properties and characteristics of the data. While the vendor provides several example configurations for how to configure the sensor, they do not behave well in all environments. To showcase this, we set up a scene of rich objects where the default parameters (and all other vendor-provided parameter sets) produce clear artifacts around depth discontinuities which could be undesirable.

We tune the depth generation for the same scene under two conditions. First, the laser emitter is enabled and dense, and high quality results are expected. In the second, the laser emitter is turned off, resulting in a difficult environment that resembles how the stereo system behaves under sunlight or at distance. These parameters are tuned directly from user preferences, without having to meticulously collect ground truth of the target scene [Sch+14].

6.2.2 Visual Tuning Results

The results of running SortCMA on tuning a stereo depth sensor for good edge performance are shown in Fig. 6.2. In both active and passive settings, we tune the algorithm with the default CMA-ES population size and perform optimization over 15 generations. Many approaches to pairwise optimization are unsuitable baselines in this context, as they depend on a reasonable reward function to model the final choice, but this reward function is unavailable.

We annotated four areas of interest for seeing differences between configurations: a large V shape, a hole in a woodcut, a close left mug handle and a further right mug handle.

For stereo matching with projected texture, the initial defaults (Fig. 6.2b) struggle with the V shape, inpaint the woodcut hole, and cannot obtain any depth inside the left mug handle. In tuning the settings (Fig. 6.2c), we were able to resolve the V shape, obtain better detail around the mug, and reject bad matches inside the woodcut, all without negatively affecting other parts of the scene. Optimization was quick: by the fourth generation, all samples had a clear V shape. By the sixth, the handle was sharply resolved. The last five generations mainly balanced density and outliers.

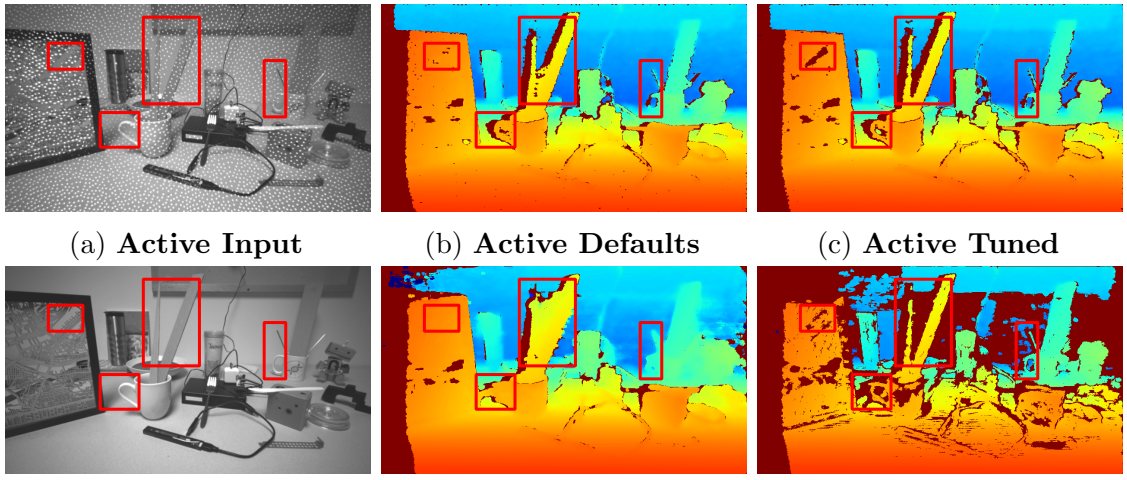
For passive stereo matching, the defaults are highly inpainted (Fig. 6.2e) across all four selected regions of interest. The tuned settings (Fig. 6.2f) resolve sharp, detailed edges at the expense of dense infill in the white background areas. In this difficult case, it took five batches to obtain a clear V shape, and about ten batches until tradeoffs were tested.

In both cases, it took about 30 minutes of tuning to get our final results, producing potentially much better configurations in a given environment. For users with particular requirements for their depth data, this could broaden and enhance the utility of commercial hardware.

6.3 Conclusion

This chapter shows how effective user preferences can be for optimization, even in a gradient-free, expensive evaluation context. In under an hour, it is possible to obtain good configurations that exhibit desirable and desirable properties (like good edge performance) that existing shipped configurations do not exhibit.

The use of a Gaussian prior for parameter optimization [Mát65] can be particularly effective for tuning of existing algorithms. In certain applications, it may be much easier to sample good algorithm configurations than to learn a reward function that models human preferences.



(a) **Active Input** (b) **Active Defaults** (c) **Active Tuned**

(d) **Passive Input** (e) **Passive Defaults** (f) **Passive Tuned**

Figure 6.2: **Tuning RealSense D435 Stereo Matching** As described in Section 6.2, we use our optimizer to tune the 35 parameters that control depth generation on a commercial depth camera. The target user preference was for more precise discontinuities. The top row demonstrates dense matching with projected texture [Kes+17; Nis84; Kon10], while the bottom row lacks texture.

Chapter 7

Learning the Value of Academic Venues

We present a method for automatically organizing and evaluating the quality of different publishing venues in Computer Science. This is largely similar to our published manuscript on the topic [Kes19], The motivation was to find an accessible, robust way to quantify academic productivity that could be easily run and reproduced (in contrast to citation metrics, which require processing all published literature for counts).

Since this method only requires paper publication data as its input, we can demonstrate our method on a large portion of the DBLP dataset, spanning 50 years, with millions of authors and thousands of publishing venues. By formulating venue authorship as a regression problem and targeting metrics of interest, we obtain venue scores for every conference and journal in our dataset. The obtained scores can also provide a per-year model of conference quality, showing how fields develop and change over time. Additionally, these venue scores can be used to evaluate individual academic authors and academic institutions. We show that using venue scores to evaluate both authors and institutions produces quantitative measures that are comparable to approaches using citations or peer assessment. In contrast to many other existing evaluation metrics, our use of large-scale, openly available data enables this approach to be repeatable and transparent.

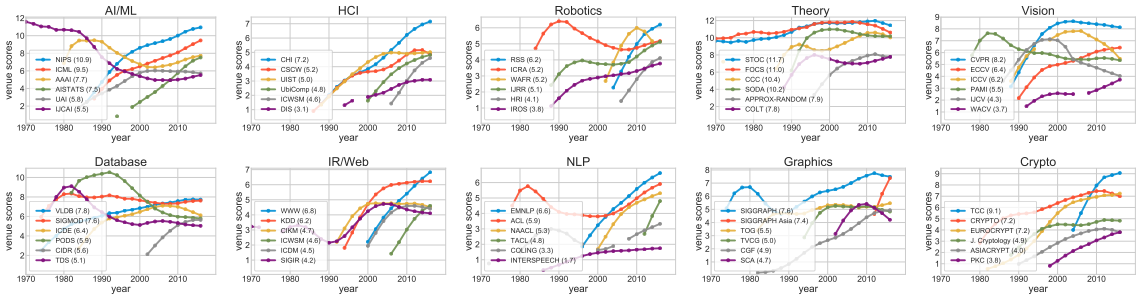


Figure 7.1: Results from our method, demonstrating differences and changes in conference quality over time in various subfields. These graphs includes size and year normalization and are the result of combining multiple different metrics of interest.

7.1 Introduction

There exist many tools to evaluate professional academic scholarship. For example Elseiver's Scorpus provides many author-level and journal-level metrics to measure the impact of scholars and their work [Col+10; SM17]. Other publishers, such as the Public Library of Science, provide article-level metrics for their published work [Fen13]. Large technology companies, such as Google and Microsoft provide their own publicly available metrics for scholarship [But11]. Even independent research institutes, such as the Allen Institute's Semantic Scholar [Amm+18], manage their own corpus and metrics for scholarly productivity. However, these author-based metrics (often derived from citation measurements) can be inconsistent, even across these large, established providers [SD18].

In this work, we propose a method for evaluating a comprehensive collection of published academic work by using an external evaluation metric. By taking a large collection of papers and using only information about their publication venue, who wrote them, and when, we provide a largely automated way of discovering not only venue's value. Further, we also develop a system for automatic organization of venues. This is motivated by the desire for an open, reproducible, and objective metric that is not subject to some of the challenges inherent to citation-based methods [SD18; BD08; GG17].

We accomplish this by setting up a linear regression from a publication record to some metric of interest. We demonstrate three valid regression targets: status as

a faculty member (a classification task), awarded grant amounts, and salaries. By using DBLP [Ley02] as our source of publication data, NSF grants as our source of awards, University of California data for salaries, and CSRankings [Ber18] for faculty affiliation status, we're able to formulate these as large data regression tasks, with design matrix dimensions on the order of a million in each dimension. However, since these matrices are sparse, regression weights can be obtained efficiently on a single laptop computer. Details of our method are explained in section 7.4.

We call our results **venue scores** and validate their performance in the tasks of evaluating conferences, evaluating professors, and ranking universities. We show that our venue scores correlate highly with other influence metrics, such as h-index [Hir05], citations or highly-influential citations [VHE15]. Additionally, we show that university rankings, derived from publication records correlate highly with both established rankings [NR18; Edu18; Ran18] and with recently published quantitative metrics [Ber18; Bla+18; Vuc+18; CAL15].

7.2 Related Work

Quantitative measures of academic productivity tend to focus on methods derived from citation counts. By using citation count as the primary method of scoring a paper, one can decouple an individual article from the authors who wrote it and the venue it was published in. Then, robust citation count statistics, such as h-index [Hir05], can be used as a method of scoring either individual authors, or a specific venue. Specific critiques of h-index scores arose almost as soon as the h-index was published, ranging from a claimed lack-of-utility [LJL06], to a loss of discriminatory power [Tol08].

Citations can also be automatically analyzed for whether or not they're highly influential to the citing paper, producing a "influential citations" metric used by Semantic Scholar [VHE15]. Even further, techniques from graph and network analysis can be used to understand systematic relationships in the citation graph [Ber07]. Citation-based metrics can even be used to provide a ranking of different universities [Bla+18].

Citations-based metrics, despite their wide deployment in the scientometrics community, have several problems. For one, citation behavior varies widely by fields [BD08]. Additionally, citations often exhibits non-trivial temporal behav-

ior [GG17], which also varies greatly by sub-field. These issues highly affect one's ability to compare across disciplines and produce different scores at different times. Recent work suggests that citation-based metrics struggle to effectively capture a venue's quality with a single number [Wal17]. Comparing citation counts with statistical significance requires an order-of-magnitude difference in the citation counts [KH17], which limits their utility in making fine-grained choices. This limitation may be present in most measures of academic productivity [Sho57]. Despite these quality issues, recent work [Vuc+18] has demonstrated that citation-based metrics can be used to build a university ranking that correlates highly with peer assessment; we show that our method provides a similar quality of correlation.

Our use of straightforward publication data (Section 7.3) enables a much simpler model. This simplicity is key, as the challenges in maintaining good citation data have resulted in the major sources of h-index scores being inconsistent with one another [SD18].

While there exist many forms of ranking journals such as Eigenfactor [Ber07] or SJR [Fal+08], these tend to focus on journal-level metrics while our work focuses on all venues, including conferences.

7.2.1 Venue Metrics

We are not the first to propose that scholars and institutions can be ranked by assigning scores to published papers[RT07]. However, in prior work the list of venues is often manually curated and assigned equal credit, a trend that is true for studies in 1990s [Gei+96] and their modern online versions [Ber18]. Instead, we propose a method for obtaining automatic scores for each venue, and in doing so, requires no manual curation of valid venues.

Previous work [YL07] has developed methods for ranking venues automatically, generating unique scores based only on author data by labeling and propagating notions of "good" papers and authoritative authors. However, this work required a manually curated seed of what good work is. It was only demonstrated to work on a small sub-field of conferences, as new publication cliques would require new labeling of "good" papers. Recent developments in network-based techniques for ranking venues have included citation information [ZW18], and are able to produce temporal

models of quality. In contrast, our proposed model doesn't require citation data to produce sensible venue scores.

Our work, in some ways, is most similar to that of CSRankings [Ber18]. CSRankings maintains a highly curated set of top-tier venues; venues selected for inclusion are given 1 point per paper, while excluded venues are given 0 points. Additionally, university rankings produced by CSRankings include a manually curated set of categories, and rankings are produced via a geometric mean over these categories. In comparison, in this work, we produce unique scores for every venue and simply sum together scores for evaluating authors and institutions.

In one of the formulations of our method, we use authors status as faculty (or not faculty) to generate our venue scores. We are not alone in this line of analysis, as recent work has demonstrated that faculty hiring information can be used to generate university prestige rankings [CAL15].

Many existing approaches either focus only on journals [Fal+08], or do not have their rankings available online. For our dataset, we do not have citation-level data available, so we are unable to compare against certain existing methods on our dataset. However, as these methods often deploy a variant of PageRank [Pag+99], we describe a PageRank baseline in Section 7.6.1 and report its results.

7.3 Data

Our primary data is the *dblp computer science bibliography* [Ley02]. DBLP contains millions of articles, with millions of authors across thousands of venues in Computer Science. We produced the results in this paper by using the dblp-2019-01-01 snapshot. We restricted ourselves to only consider conference and journal publications, skipping books preprints, articles below 6 pages and over 100 pages. We also merged dblp entries corresponding to the same conference. This lead to a dataset featuring 2,965,464 papers, written by 1,766,675 authors, across 11,255 uniquely named venues and 50 years of publications (1970 through 2019).

Our first metric of interest is an individual's status as a faculty member at a university. For this, we used the faculty affiliation data from CSRankings [Ber18], which are manually curated and contain hundreds of universities across the world and about 15,000 professors. For evaluation against other university rankings, we

used the ScholarRank [Vuc+18] data to obtain faculty affiliation, which contains a more complete survey of American universities (including more than 50 not currently included in CS Rankings). While CSRankings data is curated to have correct DBLP names for faculty, the ScholarRank data does not. To obtain a valid affiliation, the names were automatically aligned with fuzzy string matching, resulting in about 4,000 faculty with good seemingly unique DBLP names and correct university affiliation. Although those two methods are manually curated, automatic surveys of faculty affiliations have recently been demonstrated [MWC18].

Our second metric of interest was National Science Foundation grants, where we used awards from 1970 until 2018. This data is available directly from the NSF [Fou18]. We adjusted award amounts using annual CPI inflation data. We restricted ourselves to awards that had finite amount, where we could match at least half the Principal Investigators on the grant to DBLP names and the grant was above $20,000. Award amounts over 10 million dollars were clipped in a smooth way to avoid matching to a few extreme outliers. This resulted in 407,012 NSF grants used in building our model.

Our third and final metric of interest was University of California salary data [Ins18]. This was inspired by a paper that predicted ACM/IEEE Fellowships for 87 professors and used salary data [Noc+14]. We looked at professors across the entire University of California system, matching their names to DBLP entries in an automated way. We used the maximum salary amount for a given individual across the 2015, 2016 and 2017 datasets, skipping individuals making less than $120,000$ or over $800,000$ dollars. This resulted in 2,436 individuals, down from 3,102 names that we matched and an initial set of about 20,000 initial professors. As DBLP contains some Chemistry, Biology, and Economics venues, we expect that some of these are likely not Computer Science professors.

Table 7.1: Spearman's ρ correlation between rankings produced by targeting different metrics of interest.

	Faculty	NSF	Salary
Faculty	1.00	**0.91**	0.84
NSF	**0.91**	1.00	**0.86**
Salary	0.84	0.86	1.00

7.4 Method

Our basic model is that a paper in a given publication venue (either a conference or a journal), having passed peer review, is worth a certain amount of value. Certain venues are more prestigious, impactful or selective, and thus should have higher scores. Other venues have less strict standards, or perhaps provide less opportunity to disseminate their ideas [Mor+18] and should be worth less. While this model explicitly ignores the differences in paper quality at a publication venue, discarding this information enables the use of a large quantity of data to develop a statistical scoring system.

This methodology is not valid for all fields of science, nor all models of how impactful ideas are developed and disseminated. Our method requires individual authors have multiple publications across many different venues, which is more true in Computer Science than the natural sciences or humanities, where publishing rates are lower [LMS18]. If instead we assume that all research ideas produce only a single paper, or that passing peer-review is a noisy measurement of quality [Sha+18], then our proposed method would not work very well. Instead, the underlying process which supports our methodology is that good research ideas produce multiple research publications in selective venues; better ideas would produce more individual publications in higher quality venues. The concept of *All models are wrong, but some are useful* is our guide here. This assumption allows us to obtain venue scores in an automatic way, and then use these scores to evaluate both authors and institutions.

Our approach to ranking venues is to construct a regression task, apply an optimization method to solve it, and use the resulting weights. Optimization's ability to generate powerful representations is well-studied in machine learning [RHW86] and statistics [Hub85]. The resulting weights are often useful [CN11], and this methodology is widely used in natural language processing [Mik+13; PSM14].

7.4.1 Formal Setup

In general, we will obtain a score for each venue by setting up a linear regression task in the form of equation 7.1.

$$\begin{array}{c} \\ \text{auth}_1 \\ \text{auth}_2 \\ \vdots \\ \text{auth}_m \end{array} \begin{array}{c} \begin{array}{ccccc} \text{conf}_1 & \text{conf}_2 & \dots & \text{conf}_n & \end{array} \\ \begin{pmatrix} 1 & 3 & \dots & 0 & 1 \\ 1 & 0 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 2 & \dots & 4 & 1 \end{pmatrix} \end{array} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \text{isProf}_1 \\ \text{isProf}_2 \\ \vdots \\ \text{isProf}_m \end{pmatrix} \tag{7.1}$$

Authors are listed along the rows, and venues are listed along the columns. The number of publications that an author has in a conference is noted in the design matrix. There is an additional column of 1s to learn a bias offset in the regression.

Different forms of counting author credit are discussed in section 7.4.6, while different regression targets are discussed in section 7.3. In equation 7.1, the regression target is shown as a binary variable indicating whether or not that author is currently a professor. If this linear system, $Ax = b$ is solved, then the vector x will contain real-valued scores for every single publishing venue. Since our system is over-determined, there is generally no exact solution.

Instead of solving this sparse linear system directly, we instead solve a regularized regression, using a robust loss and L_2 regularization. That is, we iteratively minimize the following expression via stochastic gradient descent [RM51]

$$L(Ax, b) + \lambda ||x||^2 \tag{7.2}$$

The L_2 regularization enforces a Gaussian prior on the learned conference scores. We can perform this minimization in Python using common machine learning software packages [Ped+11]. We tend to use a robust loss function, such as the *Huber loss* in the case of regression [Hub64], which is quadratic for errors of less than δ and linear

for errors of larger than δ. It can be written as

$$L(\hat{y}, y) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{if } |y - \hat{y}| \leq \delta \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases} \tag{7.3}$$

In the case of classification, we have labels $y \in \{-1, 1\}$ and use the *modified Huber loss* [Zha04],

$$L(\hat{y}, y) = \begin{cases} \max(0, 1 - y\hat{y})^2, & \text{if } y\hat{y} \geq -1 \\ -4y\hat{y}, & \text{otherwise} \end{cases} \tag{7.4}$$

We experimented with other loss functions, such as the logistic loss, and while they tended to produce similar rankings and results, we found that the modified Huber loss provided better empirical performance in our test metrics, even though the resulting curves looked very similar upon qualitative inspection.

7.4.2 Metrics of Interest

As detailed in section 7.3, we targeted three metrics of interest: status as a faculty member, NSF award sizes, and professor salaries. Each of these metrics came from a completely independent data source, and we found that they each had their own biases and strengths (more in section 7.5).

For faculty status classification, we used the *modified huber loss* and CSRankings [Ber18] faculty affiliations. To build venue scores that reward top-tier conferences more highly, we only gave professors in the top-k ranked universities positive labels. We tried $k = 5, 16, 40, 80$, and found that we got qualitatively different results with quantitative similar performance. Unless otherwise stated, we used $k = 40$. The university ranking used to select top-k was CSRankings itself, and included international universities, covering the Americas, Europe and Asia. This classification is performed across all authors, leading to 1.7 million rows in our design matrix.

For the NSF awards, every Principal Investigator on the award had their papers up to the award year as the input features. We used a Huber loss, $\lambda = 0.03$, and experimented with normalizing our award data to have zero mean and unit variance. Additionally, we built models for both raw award sizes and log of award sizes; the raw award sizes seem to follow a power-law while the log award sizes seem distributed as approximately a Gaussian. Another model choice is whether to regress each NSF grant as an independent measurement or instead a marginal measurement which tracks the cumulative total of NSF grants received by the authors. If not all authors on the grant were matched to DBLP names, we only used the fraction of the award corresponding to the fraction of identified authors. This regression had $\sim \frac{1}{2}$ million rows in its design matrix .

For the salary data, we found that normalizing the salary data to have zero mean and unit variance led to a very poor regression result, while having no normalization produced a good result. This regression only had $\sim 2,400$ datapoints, and thus provided information about fewer venues than the other metrics of interest.

7.4.3 Modeling Change Over Time

In modeling conference values, we wanted to build a model that could adjust for different values in different years. For example, a venue may be considered excellent

in the 1980s, but may have declined in influence and prestige since then. To account for this behavior, we break our dataset into chunks of n years and create a different regression variable for each conference for each chunk. The non-temporal model is obtained simply by setting $n \geq 50$.

We also examine a model that creates an independent variable for each year, for each conference. After setting $n = 1$ in the block model, we splat each publication as a Gaussian at a given year. By modifying σ, we can control the smoothness of the obtained weights. The Gaussian is applied via a sparse matrix multiply of the design matrix A with the appropriate band diagonal sparse matrix G. The use of a truncated Gaussian (where $p < 0.05$ is clipped and the Gaussian is re-normalized) enables our matrix to maintain sparsity. We used a $\sigma = 4.5$, which produced an effective window size of about 10 years. This can be seen visually in Figure 7.2.

Our different temporal models are compared in Table 7.5 by correlating against existing author-level, journal-level and university-level metrics. For evaluation details see Section 7.6.

7.4.4 Normalizing Differences Across Years

The temporal models described in the previous section have an inherent bias. Due to the temporal window of the DBLP publication history, there is variation in distributed value due to changes annual NSF funding, the survivorship of current academic faculty, etc. To adjust for this bias, we scale each conference-year value by the standard deviation of conference values in that year. This scaling can help or hurt performance, depending on which metric of interest the model is built against. It generally produces flatter value scores over time but leads to some artifacts. The effects of this normalization are shown in Figure 7.3 and Table 7.2. In our final version, we instead normalize by the average of the top 10 conferences in each year, which produced flatter results over time,

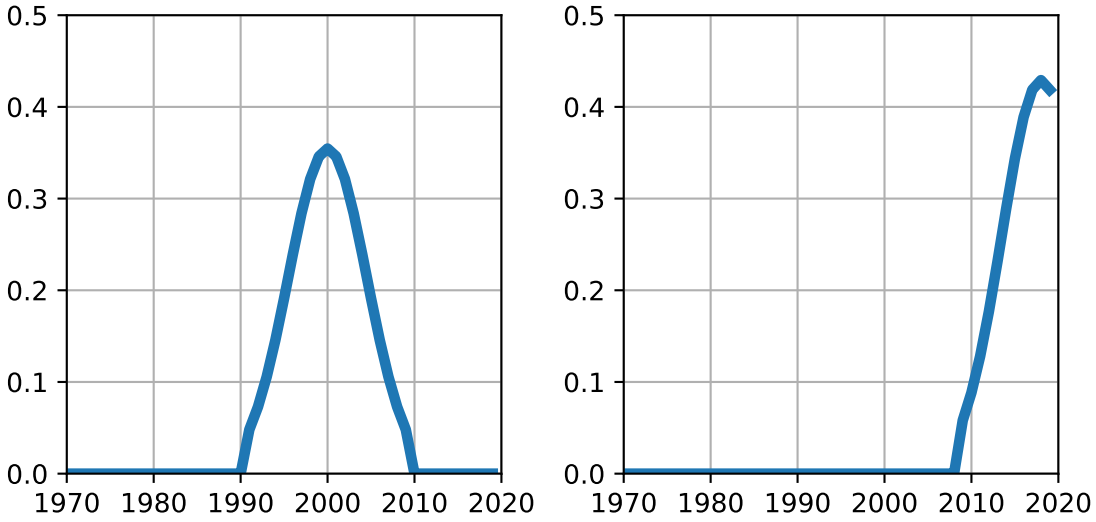


Figure 7.2: Truncated Gaussian ($\sigma = 4.5$) used to splat a publication's value across multiple years. Examples centered at the year 2000 and the year 2018

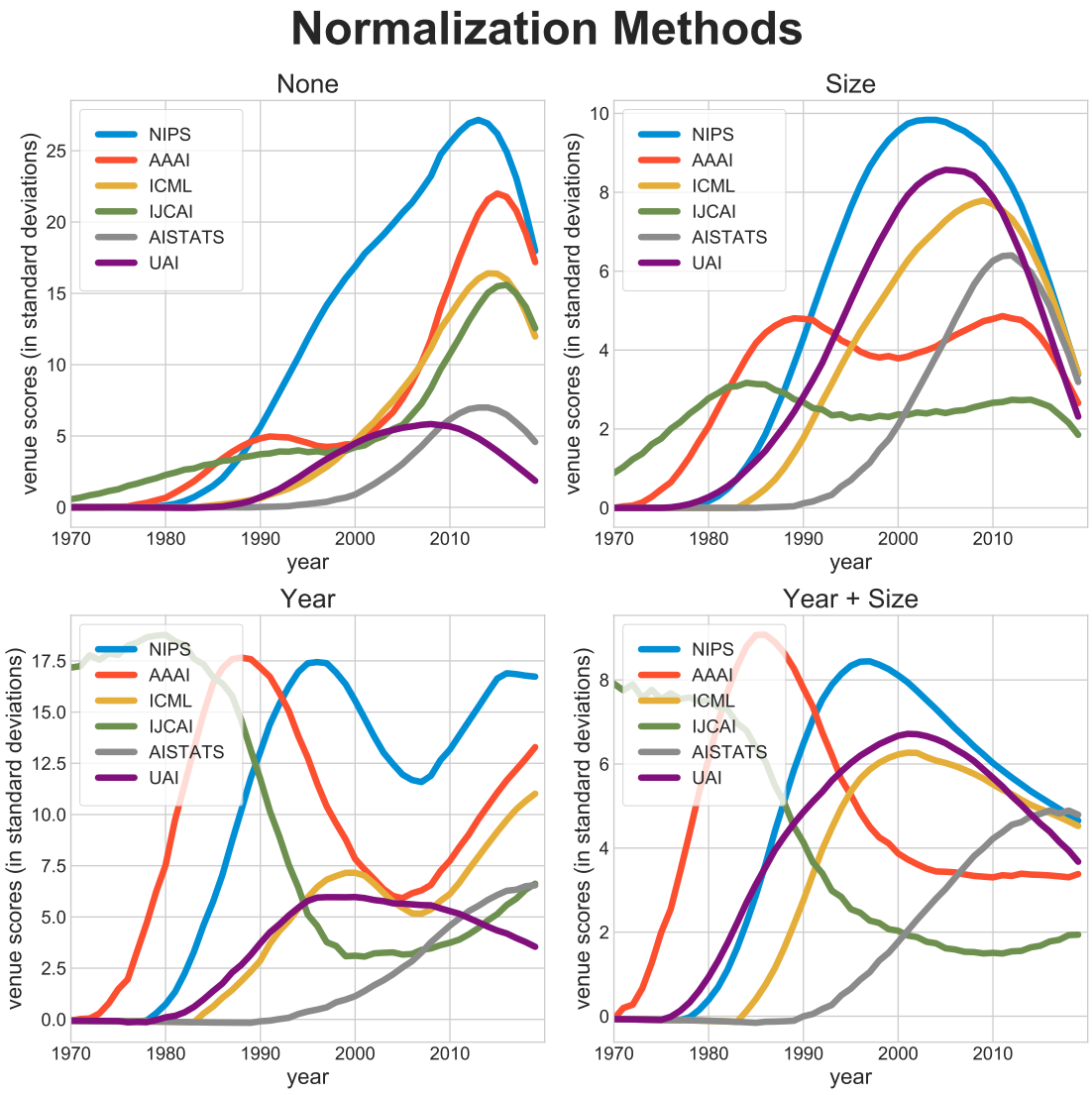


Figure 7.3: Results showing the effect of performing a normalization for venue year and size. See Sections 7.4.4 and 7.4.5.

Table 7.2: Spearman correlation between our model and existing metrics, showing the effect of different normalization schemes. See Sections 7.4.4 and 7.4.5 for model details. See Section 7.6 for evaluation details.

Normalization	influential citations (author)	h-index (author)	h-index (university)	h-index (venue)
None	0.72	0.61	0.60	**0.26**
Year	0.72	**0.66**	0.58	0.18
Size	**0.74**	0.63	**0.61**	0.25
Year + Size	0.72	0.61	0.60	**0.26**

7.4.5 Normalizing Differences In Venue Size

Our model uses L_2 regularization on the venue scores, which tends to squash the value of variables with less explanatory power.This process often resulted in the under-valuing of smaller, more selective venues. To correct for this size bias, instead of giving each paper 1 point of value in the design matrix, we give each paper $\frac{1}{M}$ credit, where M is the number of papers at that venue in that year; this step is performed before Gaussian splatting. This produced rankings which emphasized small venues and provided a good notion of efficient venues. In practice, we instead used a blended result with a multiplier of $\frac{1}{\sqrt[\alpha]{M}}$ with $\alpha = 1.5849$, the Hausdorff dimension of the Sierpinski triangle, an arbitrary constant.

7.4.6 Modeling Author Position

Another question to consider is how credit for a paper is divided up amongst the authors. We consider four models of authorship credit assignment:

1. Authors get $\frac{1}{n}$ credit for each paper, where n is the number of authors on the paper. Used by [Ber18].
2. All authors get full credit (1 point) for each paper
3. Authors receive less credit for later positions $(\frac{1}{1}, \frac{1}{2}, \frac{1}{3} \cdots \frac{1}{n})$, normalized so total

Table 7.3: Correlation (Spearman's ρ) between our model and Semantic Scholar [VHE15], showing the properties of different authorship models. For details see section 7.4.6.

		Evaluation Author Model			
		1	2	3	4
Regression Author Model	1	0.70	**0.72**	0.65	0.70
	2	0.68	**0.71**	0.61	0.67
	3	**0.71**	**0.73**	**0.66**	**0.71**
	4	0.70	**0.72**	0.65	**0.71**

credit sums to 1). This model assigns more credit to earlier authors and is used by certain practitioners [Sek08; HGC81; Hua18; Hag08].

4. The same as (3), except the last author is explicitly assigned equal credit with the first author before normalization.

Using Spearman correlation with Semantic Scholar's "highly influential citations", an evaluation metric described in depth in Sec. 7.6.4, we can evaluate each of these models. Specifically, there are two places where venues scores require a selection of authorship model. The first is how much credit is assigned to each paper when performing regression (in the case of our *faculty metric of interest*). The second is when evaluating authors with the obtained regression vector. See table 7.3 for a summary of experimental results. For the purposes of evaluation, assigning full credit to authors (model 2) produced the best results, while model 3 consistently produced the lowest quality correlations. On the other hand, for the purposes of performing the classification task, the roles are flipped. Assigning full credit (model 2) consistently produces the worst quality correlations while using model 3 produces the highest quality correlations.

7.4.7 Combining Models

Since the proposed metrics of interest (faculty status, NSF awards, salaries) were generated from different independent regression targets, with different sized design matrices, there may be value in combining them to produce a joint model. The value of ensemble models is well documented in both theory [FS97] and practice [BKV09].

Table 7.4: Correlation between our model and traditional measures of scholarly output on the dataset of CMU faculty. For model details see Section 7.4.7. For evaluation details see Section 7.6.4.

Model	citations	h-index [Hir05]	influential citations [VHE15]
Faculty	0.59	0.68	0.71
NSF	0.63	0.66	0.67
Salary	0.36	0.36	0.41
Combined	**0.69**	**0.77**	**0.75**

In the absence of a preferred metric with which to cross-validate our model, we simply perform an unweighted average of our models to obtain a *gold* model. To ensure that the weights are of similar scale, the conference scores are normalized to have zero mean and unit variance before combining them. Venue scores that are too large or too small are clipped at 12 standard deviations. For the temporal models, this normalization is performed on a per-year basis. While table 7.4 shows results for a simple combination, one could average together many models with different choices of hyperparameters, regression functions, datasets, filters to scrub the data, etc.

7.5 Results

A visual example of some of venue scores is shown in Figure 7.1. We kept the y-axis fixed across all the different Computer Science sub-disciplines to show how venue scores can be used to compare different fields in a unified metric. There are additional results in our qualitative demonstration of normalization methods, Figure 7.3.

Due to the variation in rankings produced by one's choice of hyperparameters, and the large set of venues being evaluated, we do not have a canonical set of rankings that can be presented succinctly here. Instead, we will focus on quantitative evaluations of our results in the following section.

Table 7.5: Correlation between rankings produced by our model against rankings produced by traditional scholarly metrics. Different rows correspond to different hyperparameters choices for our model. Each column is a corresponds to a traditional metric. AI = Author Highly Influential Citations, AH = Author H-index, USN = US News 2018, VH = Venue H-index, VC = Venue Citations.

Years	Metric	AI	AH	USN	VH	VC
$\sigma = 4.5$	Faculty	0.73	**0.69**	0.74	0.63	0.42
10		0.67	0.57	**0.76**	0.57	0.35
50		**0.75**	0.68	**0.76**	0.38	0.21
$\sigma = 4.5$	NSF	0.64	0.62	0.62	0.61	0.59
10		0.68	0.68	0.60	0.59	0.60
50		0.67	0.65	0.63	**0.64**	**0.67**
$\sigma = 4.5$	Salary	0.62	0.58	0.59	0.48	0.55
10		0.65	0.62	0.57	0.45	0.55
50		0.66	0.63	0.56	0.43	0.63

7.6 Evaluation

To validate the venue scores obtained by our regression methods, our evaluation consists of correlating our results against existing rankings and metrics. We consider three classes of existing scholarly measurements to correlate against: those evaluating universities, authors, and venues. Each of these classes has different standard techniques, and a different evaluation dataset, so they will be described separately in Sections 7.6.2, 7.6.4, and 7.6.3.

In the case of our proposed method, venue scores, we have a simple way to turn them from a journal-based to an author-based or institution-based metric. Venues are evaluated directly with the scores. Authors are evaluated as the dot product of venue scores and the publication vector of an author. Universities are evaluated as the dot product of venue scores and the total publication vector of all faculty affiliated with that university.

7.6.1 PageRank Baseline

Many existing approaches build on the idea of eigenvalue centrality [Ber07; YL07; ZW18]. We implemented PageRank [Pag+99] using the power iteration method to compute a centrality measure to use for both author-level and venue-level metrics. Unlike most versions of PageRank, which use citation counts, we implement two variants based solely on co-authorship information.

Author-level PageRank (PageRankA) is computed on the 1.7M x 1.7M sized co-authorship graph, where an edge is added for every time two authors co-author a paper. We found that the authors with highest centrality measures are often common names with insufficient disambiguation information in DBLP.

Journal-level PageRank (PageRankC) is computed on the 11,000 x 11,000 co-authorship graph, where an edge is added for every author who publishes in both venues. When ran on the unfiltered DBLP data, the highest scoring venue was arXiv, an expected result.

7.6.2 University Ranks

For this work, we produce university rankings simply as an evaluation method to demonstrate the quality and utility of our venue scoring system. The reader is cautioned that university ranking systems can tend to produce undesirable gaming behavior [Joh18], and are prone to manipulation.

We obtained and aligned many existing university rankings for Computer Science departments. These include rankings curated by journalistic sources, such as the US News Rankings [NR18], the QS Rankings [Ran18], Shanghai Ranking [Ran15], Times Higher Education Rankings [Edu18] and the National Research Council report [CAL15]. In addition, we consider purely quantitative evaluation systems such as ScholarRank [Vuc+18], CSRankings [Ber18], CSMetrics [Bla+18], and Prestige Rankings [CAL15]. We additionally include ScholarRank's *t10sum* across the matched faculty that our venue scores result uses.

We follow a recent paper [Vuc+18], which demonstrated the efficacy of a citation-based metric in producing rankings with large correlation against US News rankings. We extend these experiments to include more baselines. In contrast with [Vuc+18], we use a rank correlation metric (namely Kendall's τ), which naturally handles ordinal

Table 7.6: Section 7.6.2. Kendall's τ correlation across different University Rankings.

Ranking	Correlation with US News 2018
USN2018 [NR18]	1.000
USN2010 [CAL15]	0.928
Venue Scores	**0.780**
ScholarRank [Vuc+18]	0.768
ScholarRankFull	0.757
CSMetrics [Bla+18]	0.746
CSRankings [Ber18]	0.724
Times [Edu18]	0.721
NRC95 [CAL15]	0.713
t10Sum [Vuc+18]	0.713
Prestige [CAL15]	0.666
Citations [Vuc+18]	0.665
Shanghai [Ran15]	0.586
# of papers	0.585
BestPaper [Hua18]	0.559
PageRankA	0.535
PageRankC	0.532
QS [Ran18]	0.518

ranking systems. While ScholarRank [Vuc+18] claimed a correlation of > 0.9 with US News, this was under Pearson's correlation coefficient, and the result under Kendall's τ is 0.768 in the published version and 0.757 using full precision ScholarRank.

Our faculty-based regression is able to generate a result with the highest correlation against the US News rankings. We perform even better than ScholarRank, which was designed to optimize this metric (although under a non-rank correlation metric).

Table 7.7: Spearman's ρ correlation between different journal-level metrics (N=1008). For details see Section 7.6.3.

	papers	citations	h-index	PageRankC	venue scores
papers		**0.75**	0.41	**0.91**	0.25
citations	0.75		0.60	0.69	0.27
h-index	0.41	0.60		0.37	**0.65**
PageRankC	**0.91**	0.69	0.37		0.27
venue scores	0.25	0.27	**0.65**	0.27	

7.6.3 Journal-level metrics

To evaluate the fidelity of our venue scores for journals and conferences, we obtain the h-index [Hir05] and citation count for 1,308 conferences from Microsoft Academic Graph [Sch14]. We continue to use Spearman's ρ as our correlation metric, even though rank-correlation metrics can be highly impacted by noisy data [Abd90].

Under this metric, *venue scores* correlated highly with *h-index*. Notably, h-index and venue scores are highly correlated with each other, even though venue scores are much less correlated with raw conference size. See Figure 7.7 for detailed results.

7.6.4 Author-level Metrics

To evaluate our venue scores in the application of generating author-level metrics, we will use rank correlation (also known as Spearman's ρ) [Spe04] between our venue scores and traditional author-level metrics such as h-index. Google Scholar was used to obtain citations, h-index [Hir05], i10-index, and Semantic Scholar used to obtain highly influential citations [VHE15]. Prior work has critiqued the h-index measure [Yon14] and proposed an alternative metric, derived from a citation count. However, our use of a rank correlation means that monotonically transformed approximations of citation counts would lead to identical scores.

For evaluation, we collected a dataset for the largest Computer Science department in CSRankings ($n = 148$). The results are shown in Table 7.8. We can see that venue scores highly correlate with h-index, influential citations, i10-scores and CSRankings

Table 7.8: Correlation between different author-level metrics for a dataset of professors (N=148). Details are in Section 7.6.4.

	paper	cite	h	i10	CSR	venue	influence
paper		0.66	0.79	0.81	0.71	**0.94**	0.76
cite	0.66		0.93	0.88	0.49	0.66	**0.81**
hx	0.79	**0.93**		**0.97**	0.56	0.75	0.80
i10	0.81	0.88	**0.97**		0.53	0.75	0.73
CSR [Ber18]	0.71	0.49	0.56	0.53		0.84	0.64
venue	**0.94**	0.66	0.75	0.75	**0.84**		0.78
influence [VHE15]	0.76	0.81	0.80	0.73	0.64	0.78	

scores. The results from the author-based PageRank are surprisingly similar to our venue scores. However, the conference-based PageRank performed worse than venue scores on every correlation metric.

7.7 Discussion

Our results show a medium to strong correlation of venue scores to existing scholarly metrics, such as citation count and h-index. For author metrics, venue scores correlate with influential citations [VHE15] or h-index about as well as such measures correlate against each other or raw citation counts (see Table 7.8). For venue metrics, venue scores correlate with h-index (0.64) and citations (0.61) nearly as well as citations correlate with h-index (0.66). For university metrics, venue scores correlate as well with measures of peer assessment as citation-based metrics do [Vuc+18].

As h-index and citation counts have their flaws, obtaining perfect correlation is not necessarily a desirable goal. Instead, these strong correlations serve as evidence for the viability of venue scores.

Venue scores have been shown to be robust against hyperparameter choices (Tables 7.2, 7.3, 7.4, 7.5). Even venue scores produced from completely different data sources tend to look very similar (Table 7.1). Additionally, venue scores can naturally capture the variation of conference quality over time (Figures 7.1, 7.3).

As with any inductive method, venue scores are data-driven and will be subject

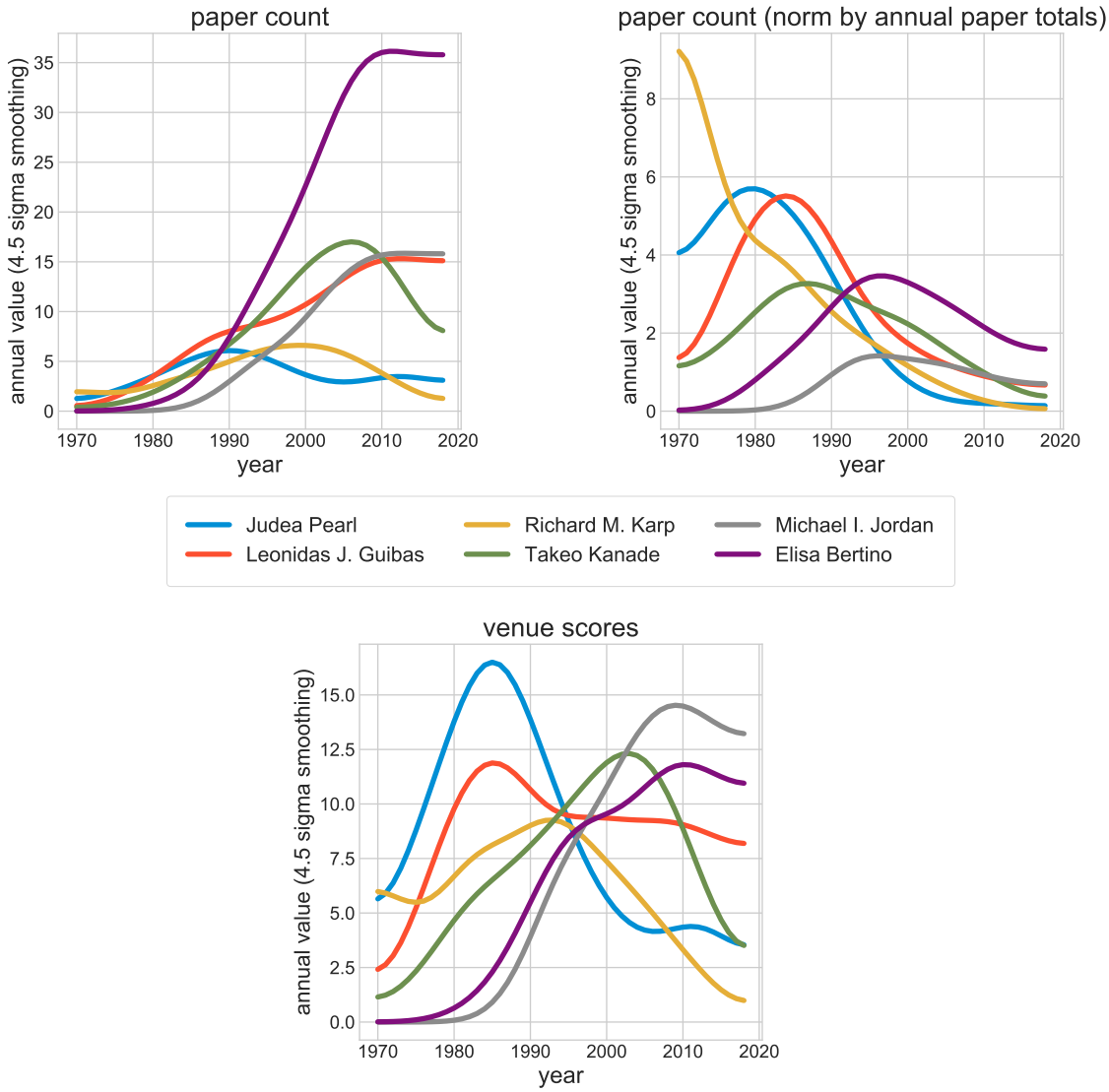


Figure 7.4: The career arcs of several accomplished Computer Scientists. The first row uses a simple model where all papers are all given equal weight; first using raw counts and then normalizing by the number of papers published each year. The second row shows our model.

to past biases. For example, venue scores can clearly be biased by hiring practices, pay inequality and NSF funding priorities. As these are the supervising metrics, bias in those datasets will be encoded in our results. For example, we found that the faculty hiring metric prioritized Theoretical Computer Science, while using NSF awards prioritized Robotics. The faculty classification task may devalue publishing areas where candidates pursue industry jobs, while the NSF grant regression task may devalue areas with smaller capital requirements. By using large datasets and combining multiple metrics in a single model (Section 7.4.7), the final model could reduce the biases in any individual dataset.

Each of our metrics of interest has an inherent bias in timescale, which our temporal normalization tries to correct for, but likely does an incomplete job of. Salaries are often higher for senior faculty. NSF Awards can have a long response time and a preference towards established researchers. Faculty classification prioritizes the productive years of existing faculty. Additionally, faculty hiring as a metric will have a bias towards work from prestigious universities [CAL15] and their venue preferences. Some of these issues also exist in citation metrics, and may be why our uncorrected models correlated better with them (Table 7.2).

7.8 Similarity Metrics

While the previous sections of this paper have focused on evaluation, the same dataset can be used to organize venues into groups. For organization, we use a much smaller dataset, using data since 2005 and only evaluating the 1,155 venues that have at least 20 R1 universities with faculty publishing in them. We then build the venue $\times$ author matrix, counting the number of papers each that author published in each venue. Performing a d-dimensional Latent Dirichlet Allocation [BNJ03], we obtain a 50-dimensional vector representing each conference in a meaningful way.

These vectors can then be clustered [AV07] to produce automatic categories for each conference. These high dimensional vectors can also be embedded [MH08] into two dimensions to produce a visual map of Computer Science. See Figure 7.5 for our result. These clusters represent natural categories in Computer Science. For example, it is easy to see groups that could be called Theory, Artificial Intelligence, Machine Learning, Graphics, Vision, Parallel Computing, Software Engineering,

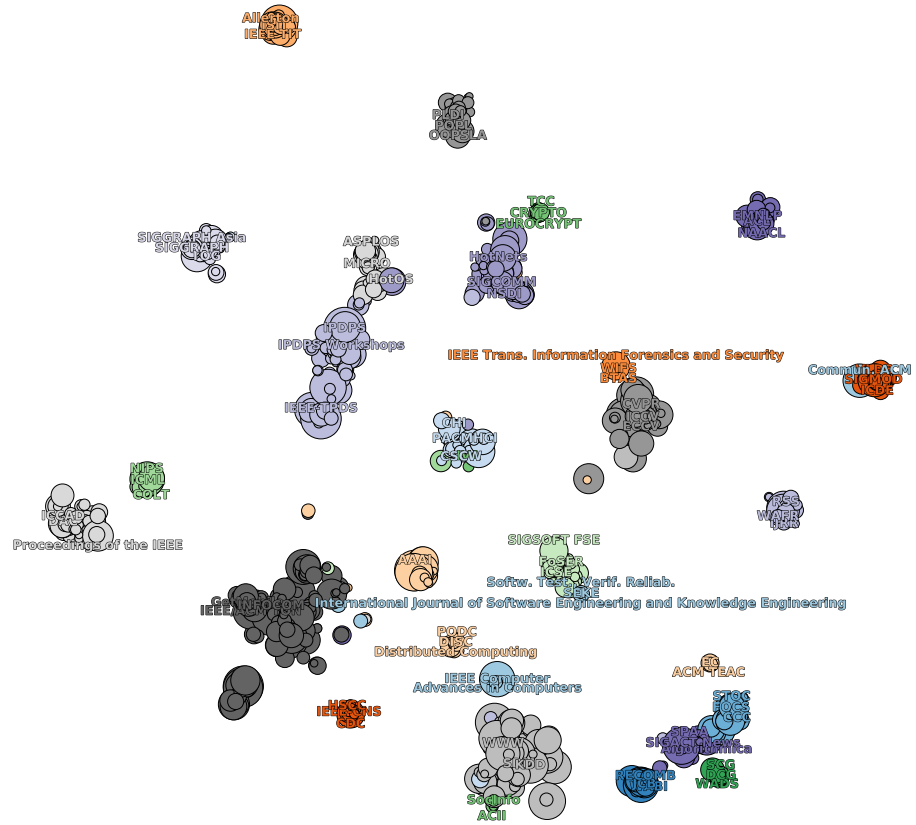


Figure 7.5: Automatic clustering for venues in Computer Science, the largest venues in each cluster are labeled.

Human-Computer Interaction, among many others.

While some clusters are distinct and repeatable, others are not. When datasets contain challenging cases, ideal clustering can be hard to estimate [Ben15]. Using silhouette scores [Rou87], we can estimate how many natural clusters of publishing exist in Computer Science. In our experiments, silhouette scores were maximized with 20 to 28 clusters. As the clustering process is stochastic, we were unable to

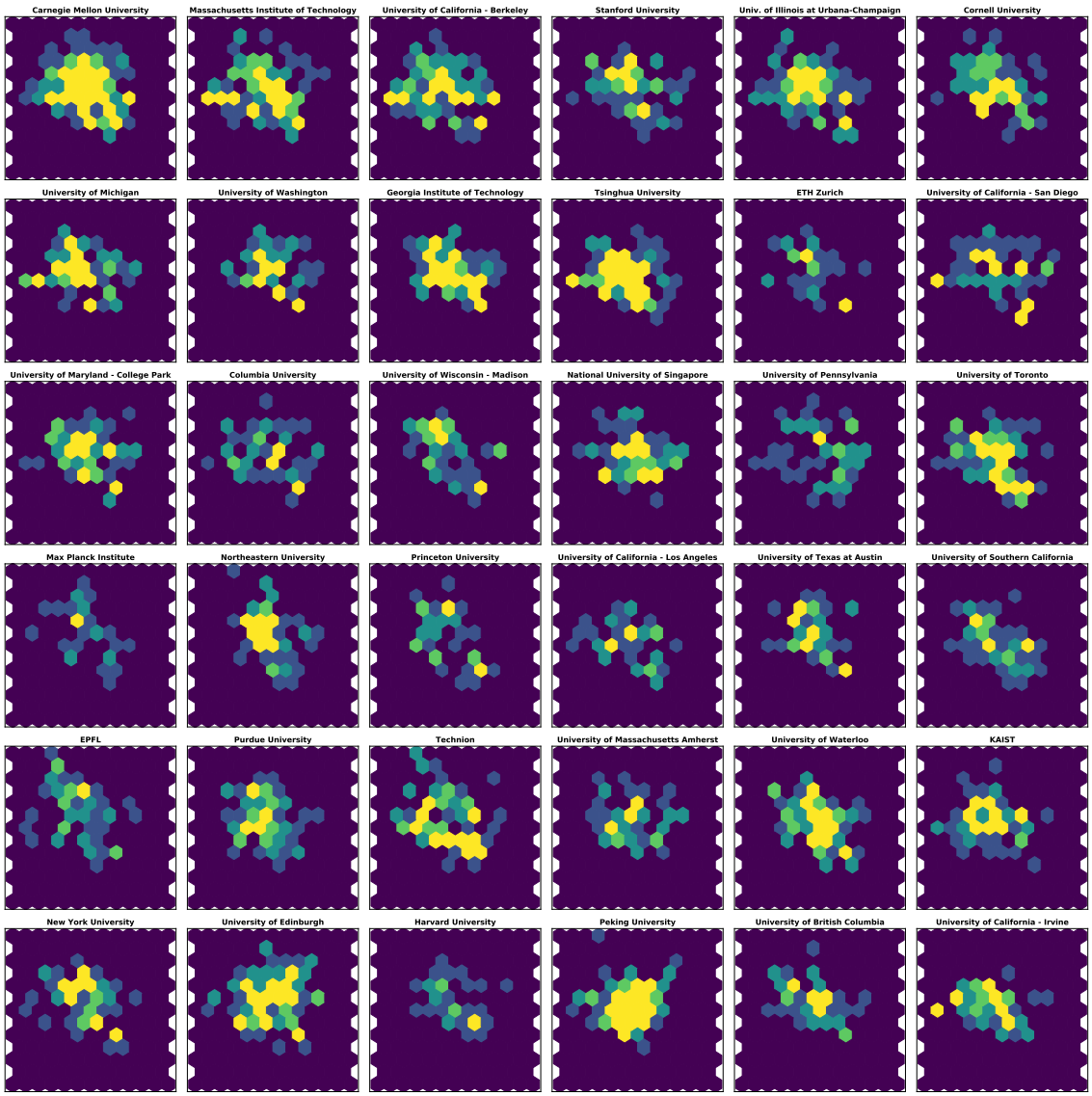


Figure 7.6: Heatmap showing the differences in research focus across different universities. Figure 7.5 can be used as a guide.

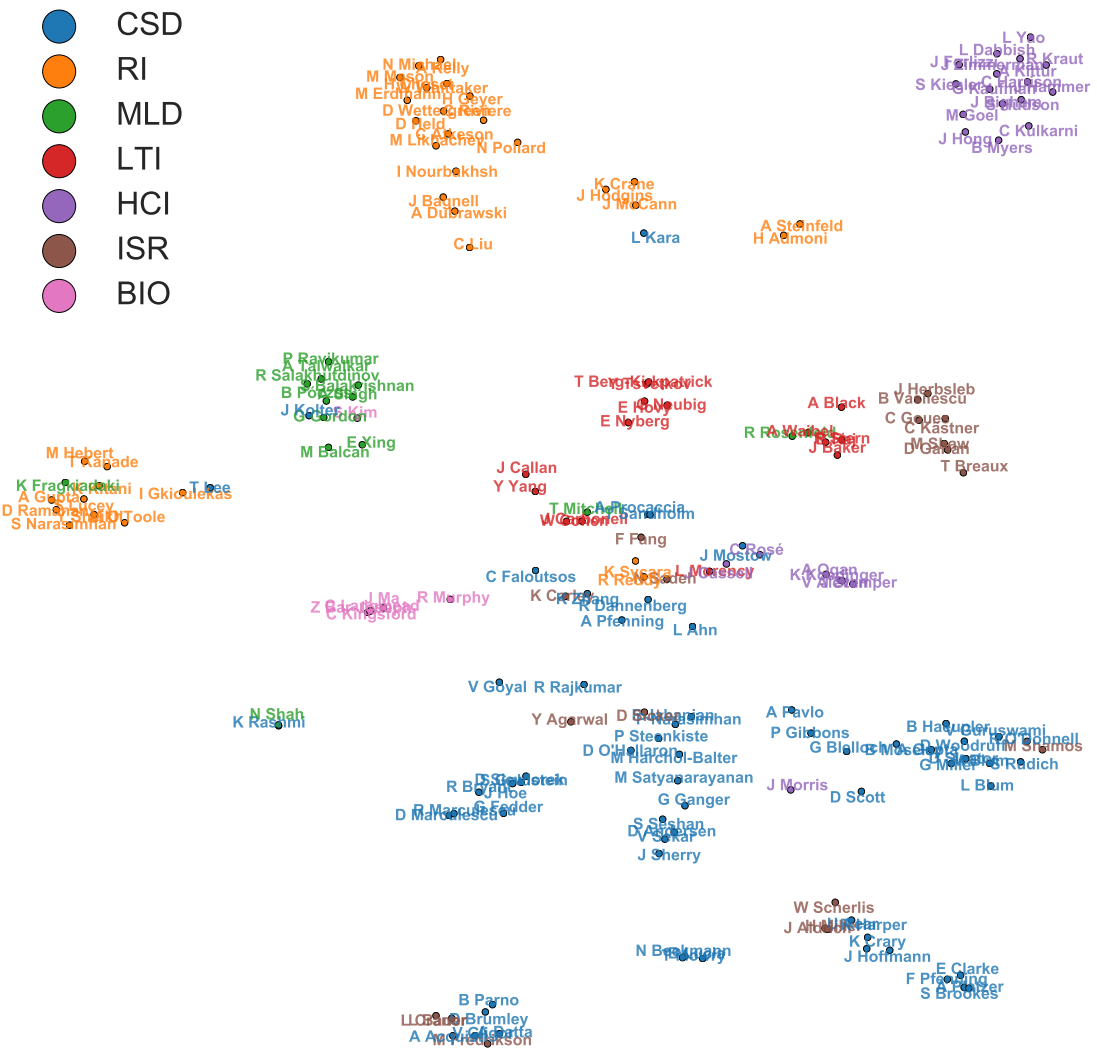


Figure 7.7: An embedding of Carnegie Mellon University's School of Computer Science with colors indicating sub-departments. For example, the Robotics Institute (RI) has clear clusters for Robotics, Graphics, CV and HRI.

determine the optimal cluster number with statistical significance.

By embedding each author with the weighted average of their publication's vectors, we can also obtain a *fingerprint* that shows which areas of Computer Science each university focuses on. See Figure 7.6 for an example of such fingerprints for many top departments. The same clustering method can be used to analyze the focus areas of a single department, for an example see Figure 7.7.

7.9 Conclusion

We have presented a method for ranking and organizing a scholarly field based only on simple publication information- namely a list of papers, each labeled with only their published venue, authors, and year. By regressing venue scores from metrics of interest, one obtains a plausible set of venue scores. These scores can be compared across sub-fields and aggregated into author-level and institution-level metrics. The scores provided by this system, and their resulting rankings, correlate highly with other established metrics. As this system is based on easily obtainable, publicly available data, it is transparent and reproducible. Our method builds on simple techniques and demonstrates that their application to large-scale data can produce surprisingly robust and useful tools for scientometric analysis.

7.10 Credit Assignment

In order to address issues of collinearity raised by having authors who publish papers together, we wanted to solve a credit assignment problem. We adapted a well-known method for addressing this by adapting regularized plus minus [Sil10] from the sports analytics literature. In our case, we simply regress each publications values from its authors, as shown below.

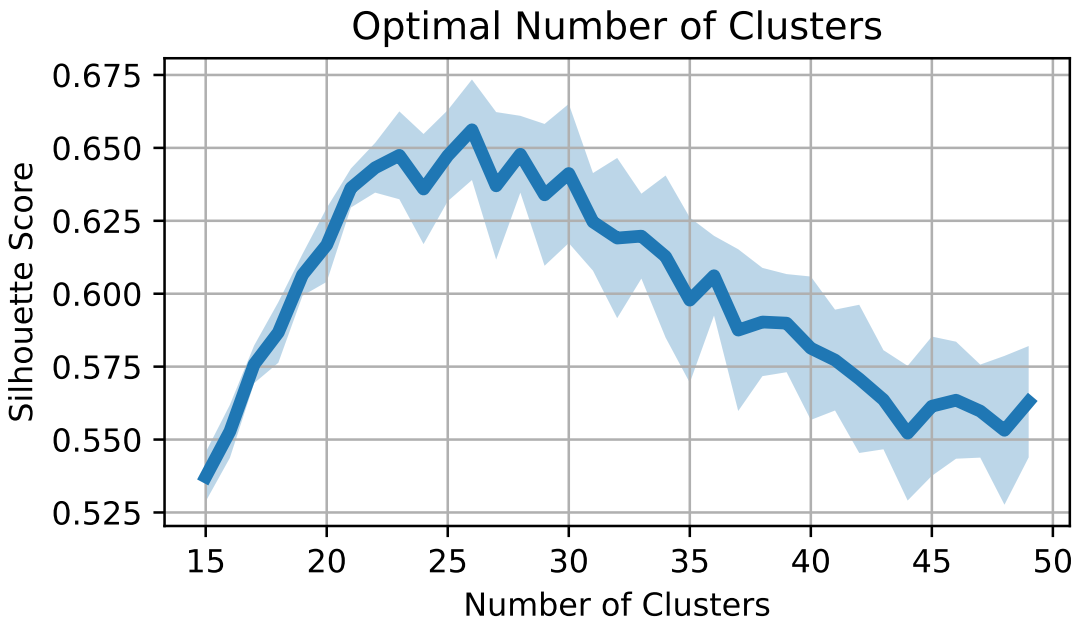


Figure 7.8: Curve showing the natural number of clusters in the dataset using R1 authors

$$\begin{array}{c} \\ \text{paper}_1 \\ \text{paper}_2 \\ \vdots \\ \text{paper}_m \end{array} \begin{array}{c} \begin{array}{cccc} \text{author}_1 & \text{author}_2 & \dots & \text{author}_n \end{array} \\ \begin{pmatrix} 1 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & 1 \end{pmatrix} \end{array} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \text{score}_1 \\ \text{score}_2 \\ \vdots \\ \text{score}_n \end{pmatrix} \tag{7.5}$$

This technique produced scores that correlated highly with total value scores. While this may be valuable for understanding an individual's contribution, but we were unable to find an evaluation for this method. The form of this model that considers n-wise or pairwise relationships was also considered, but these matrices were too large for us to run trials.

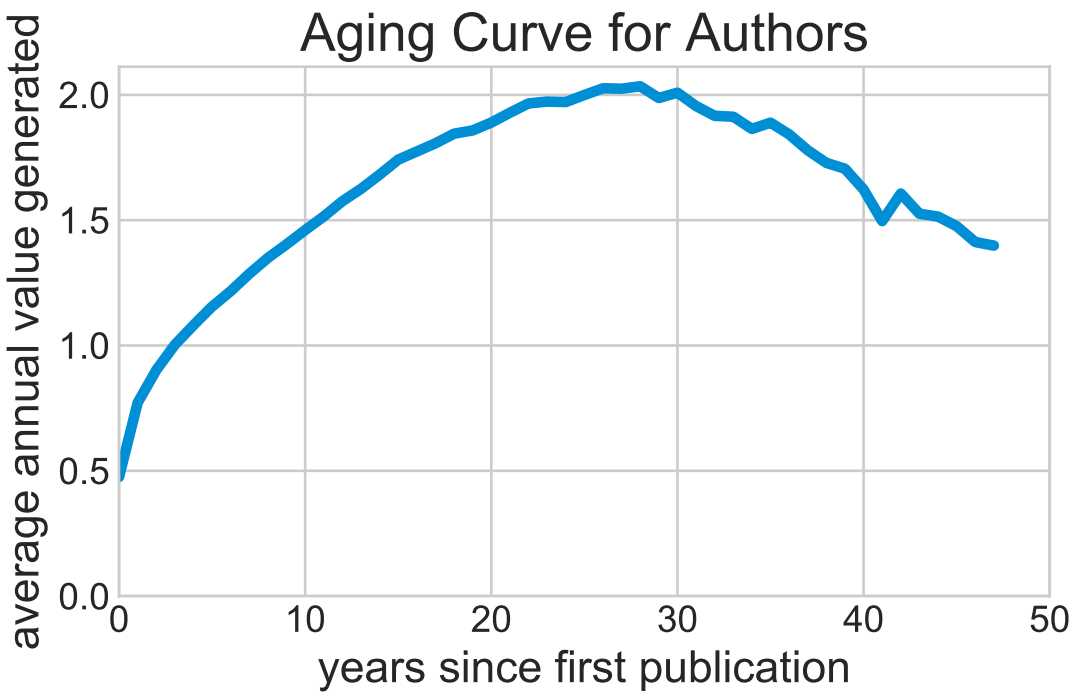


Figure 7.9: The average productivity of all DBLP authors for that year of their publishing career.

7.11 Aging Curve

To evaluate if our model makes a sensible prediction over the timescale of a scholar's career, we built a model to see what an average academic career looks like, given that the author is still publishing in those years. See Figure 7.9. Our model suggests a rise in productivity for the first 20 years of one's publishing history, and then a steady decline.

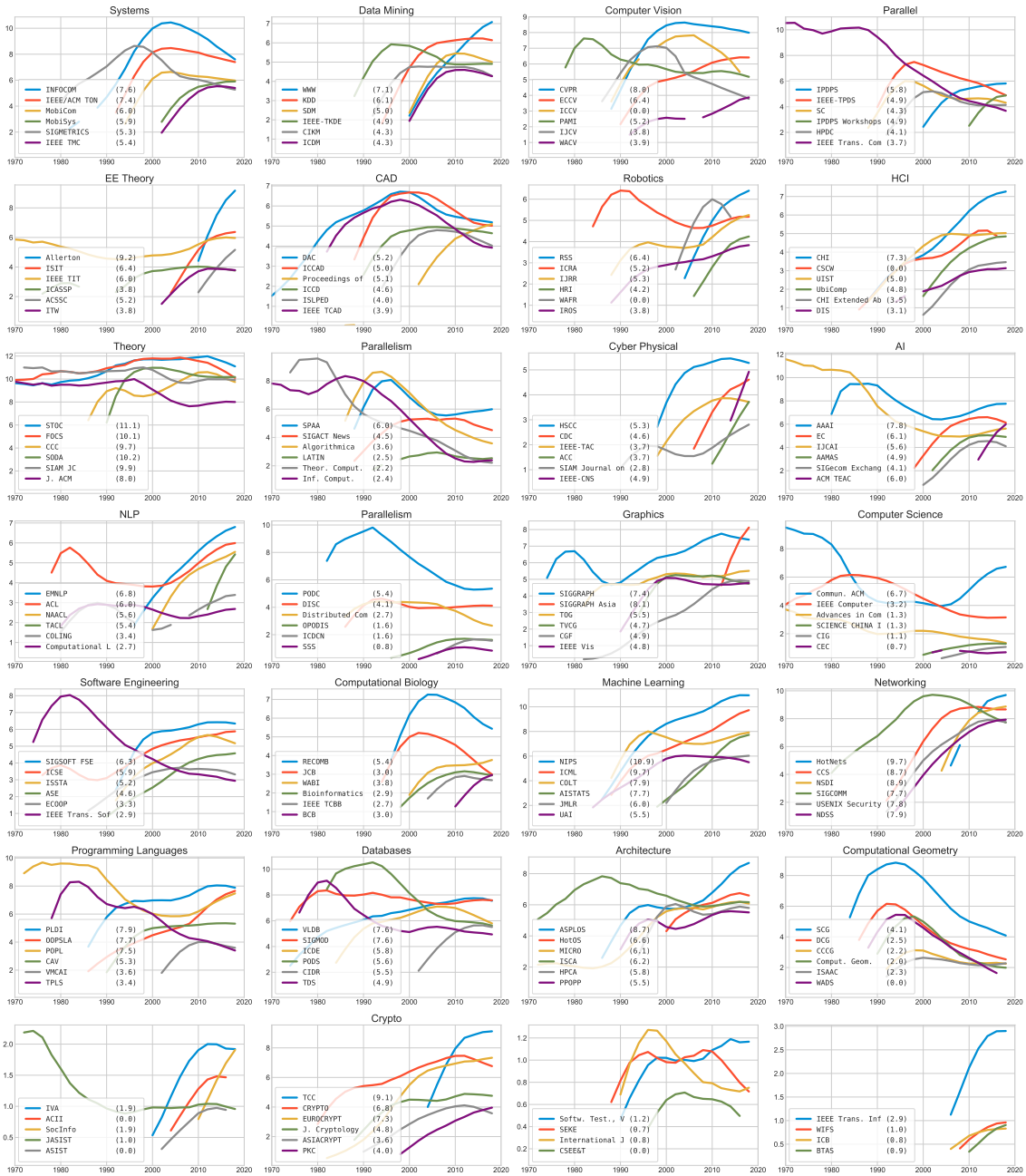


Figure 7.10: An automatic clustering and rating of the entirity of CS.

Table 7.9: Top Venues

Name	Score	Size
STOC	11.71	128
FOCS	11.04	120
NIPS	10.94	484
Conference on Computational Complexity	10.41	53
SODA	10.17	224
SIAM J. Comput.	9.98	138
HotNets	9.53	46
ICML	9.46	303
TCC	9.06	76
NSDI	8.76	69
CCS	8.76	114
INFOCOM	8.59	434
ITCS	8.56	103
Allerton	8.53	331
ASPLOS	8.43	48
SIGCOMM	8.31	55
CVPR	8.22	606
PLDI	8.05	68
J. ACM	8.03	82
Theory of Computing	7.94	29
USENIX Security Symposium	7.91	62
NDSS	7.86	63
APPROX-RANDOM	7.86	74
COLT	7.78	82
VLDB	7.75	185
IEEE/ACM Trans. Netw.	7.74	226
AAAI	7.73	403
SIGMOD Conference	7.62	98
SIGGRAPH	7.59	125
AISTATS	7.52	138

Table 7.10: Top Universities

school	authors	papers	venue score	size normed
Carnegie Mellon University	174	17275	73126	14159
University of California - Berkeley	108	11011	51062	10884
Massachusetts Institute of Technology	108	9613	47037	10026
Univ. of Illinois at Urbana-Champaign	103	11248	44714	9627
Technion	96	8795	39601	8656
Stanford University	69	7028	36169	8513
Georgia Institute of Technology	108	9337	38083	8118
Tsinghua University	150	14643	40662	8104
University of California - Los Angeles	46	6589	30368	7888
University of Michigan	83	7897	33666	7598
Tel Aviv University	48	5468	28816	7404
University of California - San Diego	74	6646	30836	7142
University of Maryland - College Park	76	7751	30795	7089
ETH Zurich	39	6673	26120	7081
Cornell University	81	5918	30278	6871
University of Washington	69	5727	29087	6846
University of Southern California	58	7206	26044	6387
Columbia University	53	5064	25251	6330
EPFL	55	6927	25320	6290
Princeton University	47	4991	24203	6252
HKUST	57	6640	25134	6190
National University of Singapore	75	7210	25123	5801
University of Pennsylvania	53	5340	22696	5690
University of California - Irvine	71	6624	24070	5628
Pennsylvania State University	49	5668	21325	5451
Peking University	147	11858	27050	5413
University of Toronto	99	5955	24759	5376
University of Texas at Austin	52	4485	21225	5346
University of California - Santa Barbara	38	4698	19137	5224
University of Waterloo	105	7316	23781	5099
Max Planck Institute	32	4307	17808	5093

Table 7.11: Top Authors

Name	Score
Philip S. Yu	4354
H. Vincent Poor	3873
Kang G. Shin	3540
Jiawei Han 0001	3143
Micha Sharir	3086
Thomas S. Huang	2992
Don Towsley	2913
Xuemin Shen	2755
Luc J. Van Gool	2700
Noga Alon	2596
Christos H. Papadimitriou	2531
Leonidas J. Guibas	2525
Mahmut T. Kandemir	2467
Jie Wu 0001	2452
Wen Gao 0001	2416
Shuicheng Yan	2373
Rama Chellappa	2286
Alberto L. Sangiovanni-Vincentelli	2266
Georgios B. Giannakis	2266
Yunhao Liu	2257
Mohamed-Slim Alouini	2238
Michael I. Jordan	2213
Christos Faloutsos	2173
Massoud Pedram	2153
Dacheng Tao	2149
Hans-Peter Seidel	2145
Luca Benini	2139
Moshe Y. Vardi	2092
Lajos Hanzo	2081
Yishay Mansour	2073

Chapter 8

Additional Results

This chapter briefly notes some otherwise unpublished work that did not make a full length conference publication.

8.1 Alternative Rendering Formulations

We experimented with different alternatives to 3D Gaussians as an underlying shape representation. For example, surfels [Pfi+00] with infinite plane intersections, and configurable fuzz (Fig. 8.1). The confiurable fuzz was a sigmoid function based on intersection distance from the surfel center. While these sometimes provided sharper details, they didn't optimize as well as the 3D Gaussian formulation described above.

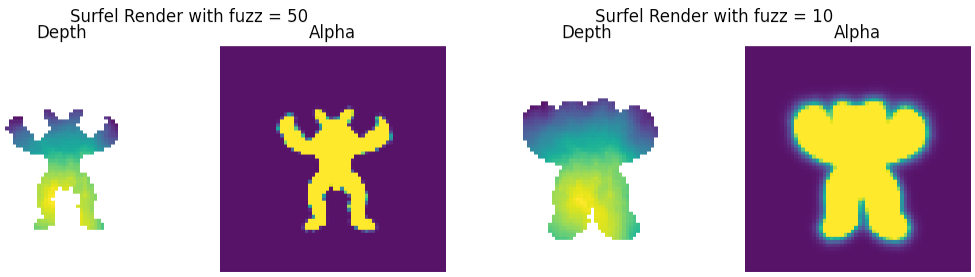


Figure 8.1: **Surfel Renderer.**

Another idea that we pursued was implementing a differentiable render for Varia-

tional Implicit Surfaces [TO99], also known as Gaussian Process Surfaces [WF07]. Here, we used a similar exponential blending trick as the one used in Fuzzy Metaballs, where the two terms were distance along the ray (to handle occlusion) and distance orthogonal to the ray (to handle priority) compared to all the vertex control points. Forward passes could be further refined by running gradient descent along the ray to minimize SDF distance. Despite our best efforts at linearization, reparameterization, we never got the shape optimization to work well.

The basic shape parameterization we experiment with was a simple squared distance kernel over control points.

$$f(x) = \sum_{i}^{N} \lambda_i ||\mu_i - x||_2$$

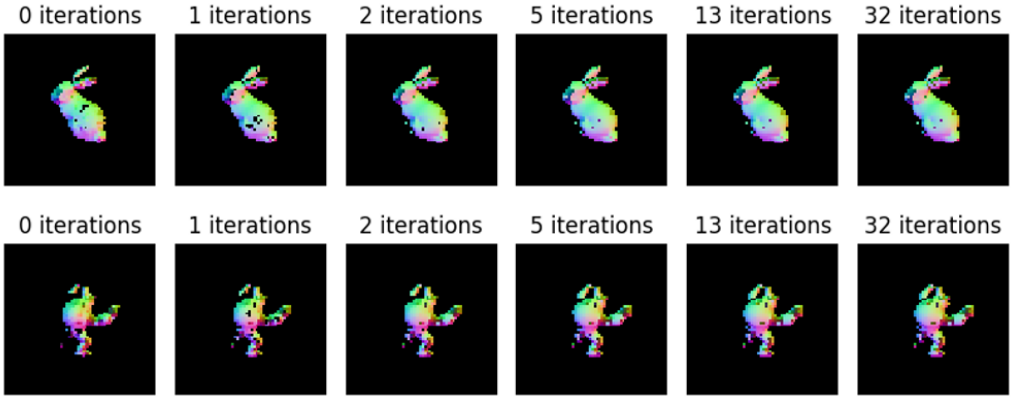


Figure 8.2: **Variational Implicit Surface Renderer Forward Pass.** Using 256 control points. Based on either the exponential weighting trick to get initialization, or further steps of SDF improvement. Normals based on instantaneous gradient are shown.

8.1.1 Multivariate Logistic

We experimented with an alternative primitive, which seems to behave like a generalized simplex (triangle in two dimensions, tetrahedron in three dimensions). This has a closed form cumulative distribution function, and is one potential multivariate

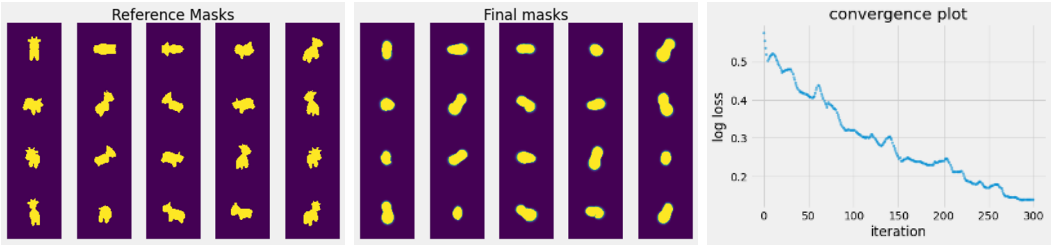


Figure 8.3: **Variational Implicit Surface Renderer Shape From Silhouette.** Despite having about 100 times more runtime than the similar Fuzzy Metaballs experiment, there is marginal optimization of the shape towards the desired object.

generalization of the logistic distribution. We were hoping this could enable other renderers, and it might be an interesting replacement primative for the smooth, defined everywhere optimization.

$$w(x;\mu,\Sigma) = \Sigma(x-u) \qquad z^T\Sigma z \geq 0$$

$$f(x;\mu,\Sigma) = D! \frac{e^{-\sum_j^D w_j}}{|\Sigma|\left(1+\sum_j^D e^{-w_j}\right)^{D+1}}$$

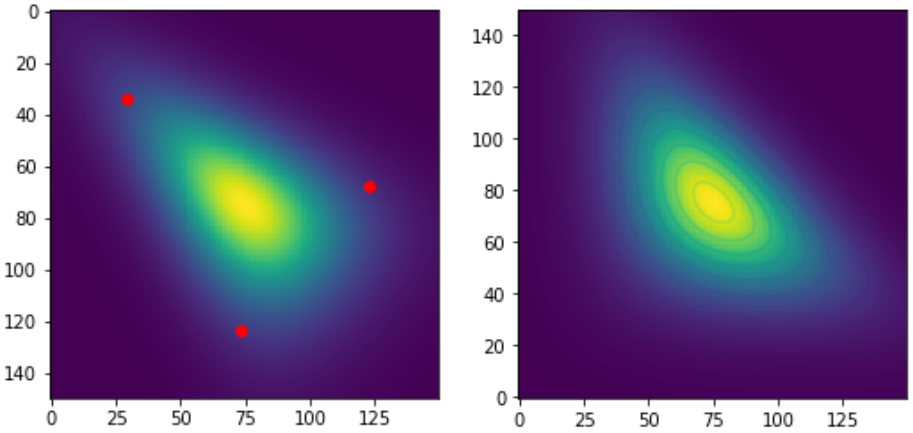


Figure 8.4: **Multivariate Logistic Primitive.**

It seems to be slightly different than the typical multivarariate logistic distribution used by others [Fra04; MO90; Arn96; FX89; McD09; MA+73; Bal91].

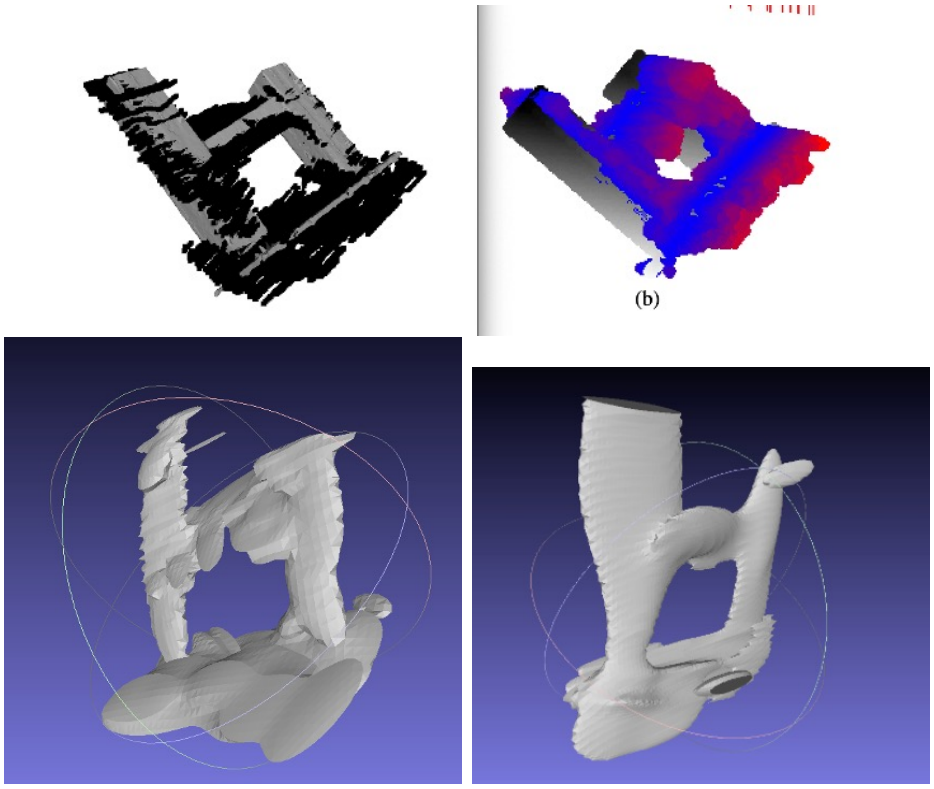


Figure 8.5: **Sonar Results.** The first row shows baseline results for 14 degree aperture reconstructions. Point cloud based on maximal ray values on the left, [WGK20] on the right. The second row shows the resulting Gaussian reconstructions for a 14 degree and 28 degree aperture. The original work didn't show visual results for 28 degree data, so it is not available for comparison.

8.2 Sonar Results

We briefly experimented with using 3D Gaussians as an underlying representation for sonar reconstructions. Of note, we compare against the results shown in [WGK20] for both 14 degree aperture and 28 degree aperture sonar. We develop a strong point cloud baseline, and show that we can get comparable results by using the dense sonar returns to optimized a Gaussian Mixture Model likelihood in JAX.

Chapter 9

Conclusions

In Freeman Dyson's characterization [Dys09], I believe this thesis was largely done in the manner of a *frog*, focusing on very specific problems and providing good solutions to them. Stepping back, there is a clear theme of efficiency, generality across domains, and tiny bits of mathematics to ground the work. Often, these specific approaches used Gaussians, and hence the title of the thesis. But overall, trying to take a look as a bird, the goals were about efficiency, generality and robustness in intelligent methods that work with real world data. I hope the themes will continue in my future work, but which problem domains they will focus on seems hard to see at this moment.

Looking to the future, there are potentially other massive gains in efficiency that could be possible for techniques in computer vision and robotics. For example, building neural network back-ends that efficiently process videos for learned ML inference. Or perhaps building methods that efficiently learn robot policies from a small dataset of learned trajectories and the appropriate additional information. If we formulate solutions directly for these problems instead of copying approaches used elsewhere, perhaps we can see the same benefits as 3D Gaussians did for differentiable rendering.

However, I will outline some specific directions that would be interesting in the space of 3D reconstruction. A rough concept is that, having built a differentiable ray-tracer, we could compute lighting bounces, and perhaps reconstruct HDR environment maps or perform per-Gaussian material capture for relighting purposes.

9.1 Bootstrapping Solutions

Since Fuzzy Metaballs are extremely fast, simple and converge well, they may serve as an ideal *coarse* method for quickly seeding a solution for a *fine* method. This would successfully demonstrate how low degree of freedom models can integrate with, and provide comparable solutions to, higher quality models that can take days to optimize.

In this case, we were thinking of tackling the reconstruction of articulated bodies from a monocular video [Yan+21a; Yan+21b; Yan+22; Yan+23]. While prior work uses mesh or NeRF-based differentiable rendering, we believe using Fuzzy Metaballs might greatly benefit these approaches. Mesh differentiable renderers struggle to converge well in our experiments without careful regularization and tuning, and are significantly slower. While prior work solves these issues by doing multiple stages of coarse-to-fine solving and constantly remeshing, we hope that doing an initial solve with an implicit surface and then a single pass of high-quality optimization with a high resolution mesh.

Another potential application of this might be cases where users currently deploy NeRF-based models, such as in scene reconstruction [Suc+21]. This may help alleviate the need for COLAMP [Sch+16] (or other SfM methods) before switching to optimization with differentiable rendering.

9.2 Better Applications

Fuzzy Metaballs (Chapter 2) are extremely low dimensional and the low dimensionality acts as an implicit regularizer (Section 2.10). Compared to other methods, they are extremely parameter efficient (Section 2.6).

While the focus of the thesis has thus far been on standard computer graphics datasets (such as the Stanford Bunny) and standard computer vision style inputs (cell phone videos), we would like to extend our method to other domains. Namely, we would like to demonstrate the benefits of low degree of freedom models in highly noisy and difficult tasks.

We have identified several possible areas where such regularization might be beneficial: in non-light-of-sight imaging [TSG19], lightcurve inversion [KT01] and

CryoEM [BPF15]. In all these contexts, getting good imaging information is extremely hard and low degree of freedom models could be desirable. Often the measurements are based on some aggregated sensing (e.g. all the sonar returns along a potential arc, all the light coming from an asteroid) and the reconstruction is a very difficult inverse problem. In some of these cases, getting alternative information is very expensive. The alternative to CryoEM is getting chemicals to form crystal structures and using NMR techniques. The alternative to lightcurve inversion for asteroids is to launch a spacecraft for a flyby.

In these applications, we expect that a differentiable renderer using implicitly regularized, low-degree-of-freedom models would have large benefits.

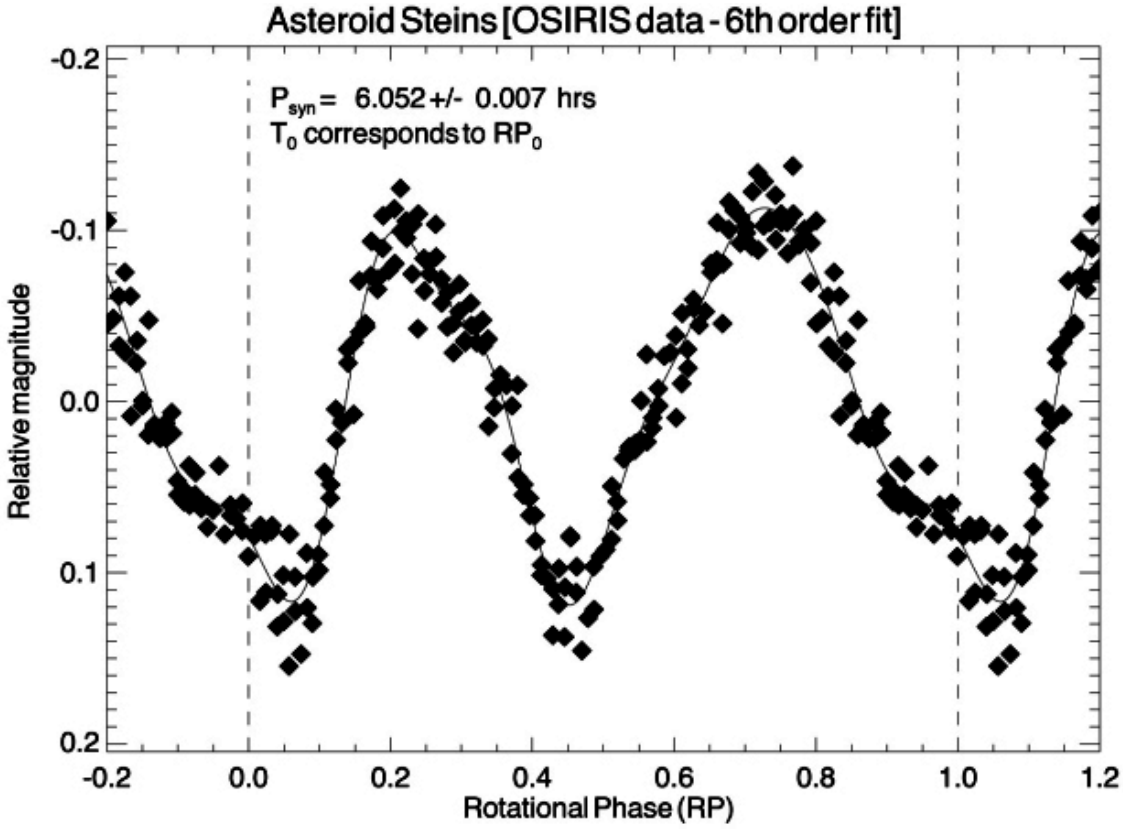


Figure 9.1: Asteroid Lightcurve

Thank you for the kind readership. Many have supported me during the writing and revision of this thesis, but all errors are my own.

Leonid Keselman, September 2023

Bibliography

[Abd90] Abdullah, Mokhtar Bin. "On a Robust Correlation Coefficient". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 39.4 (1990), pp. 455–460. ISSN: 00390526, 14679884. URL: http://www.jstor.org/stable/2349088 (pg. 133).

[Adr+22] Adriaensen, Steven, Biedenkapp, André, Shala, Gresa, Awad, Noor, Eimer, Theresa, Lindauer, Marius, and Hutter, Frank. *Automated Dynamic Algorithm Configuration*. 2022. DOI: 10.48550/ARXIV.2205.13881. URL: https://arxiv.org/abs/2205.13881 (pg. 88).

[Agi72] Agin, Gerald Jacob. "Representation and Description of Curved Objects". PhD thesis. Stanford University, 1972. URL: https://apps.dtic.mil/sti/citations/AD0755139 (pg. 2, 8, 46).

[ALD06] Adams, Bart, Lenaert, Toon, and Dutré, Philip. *Particle Splatting: Interactive Rendering of Particle-Based Simulation Data*. Report CW 453. KU Leuven, July 2006, p. 18. URL: http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW453.abs.html (pg. 9, 46).

[Amm+18] Ammar, Waleed et al. "Construction of the Literature Graph in Semantic Scholar". In: *NAACL HLT*. New Orleans - Louisiana: Association for Computational Linguistics, 2018, pp. 84–91. DOI: 10.18653/v1/N18-3011. URL: http://aclweb.org/anthology/N18-3011 (pg. 115).

[Ans+14] Ansel, Jason, Kamil, Shoaib, Veeramachaneni, Kalyan, Ragan-Kelley, Jonathan, Bosboom, Jeffrey, O'Reilly, Una-May, and Amarasinghe, Saman. "OpenTuner: An Extensible Framework for Program Autotuning". In: *International Conference on Parallel Architectures and Compilation Techniques*. Edmonton, Canada, Aug. 2014. URL: http://groups.csail.mit.edu/commit/papers/2014/ansel-pact14-opentuner.pdf (pg. 87, 90).

[Ans+15] Ansótegui, Carlos, Malitsky, Yuri, Samulowitz, Horst, Sellmann, Meinolf, and Tierney, Kevin. "Model-Based Genetic Algorithms for Algorithm Configuration". In: *Proceedings of the 24th International Conference on*

Artificial Intelligence. IJCAI'15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 733–739. ISBN: 9781577357384 (pg. 88).

[Ans+21] Ansotegui, Carlos, Sellmann, Meinolf, Shah, Tapan, and Tierney, Kevin. *Learning How to Optimize Black-Box Functions With Extreme Limits on the Number of Function Evaluations*. 2021. DOI: 10.48550/ARXIV.2103.10321 (pg. 88, 97, 109).

[Arn96] Arnold, Barry C. "Distributions with Logistic Marginals and/or Conditionals". In: *Lecture Notes-Monograph Series* 28 (1996), pp. 15–32. ISSN: 07492170. URL: http://www.jstor.org/stable/4355881 (pg. 149).

[AV07] Arthur, David and Vassilvitskii, Sergei. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. New Orleans, Louisiana: SIAM, 2007, pp. 1027–1035. ISBN: 978-0-898716-24-5 (pg. 72, 136).

[Bak+07] Baker, Simon, Roth, Stefan, Scharstein, Daniel, Black, Michael J., Lewis, J.P., and Szeliski, Richard. "A Database and Evaluation Methodology for Optical Flow". In: *International Conference on Computer Vision*. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4408903 (pg. 54).

[Bal91] Balakrishnan, Narayanaswamy. *Handbook of the logistic distribution*. CRC Press, 1991 (pg. 149).

[BD08] Bornmann, Lutz and Daniel, Hans-Dieter. "What do citation counts measure? A review of studies on citing behavior". In: *J. Doc.* 64.1 (2008), pp. 45–80 (pg. 115, 116).

[Bel+61] Bell, C. G., Fujisaki, H., Heinz, J. M., Stevens, K. N., and House, A. S. "Reduction of Speech Spectra by Analysis-by-Synthesis Techniques". In: *The Journal of the Acoustical Society of America* 33.12 (1961), pp. 1725–1736. DOI: 10.1121/1.1908556 (pg. 11).

[Ben15] Ben-David, Shai. "Clustering is Easy When ....What?" In: *arXiv e-prints*, arXiv:1510.05336 (Oct. 2015), arXiv:1510.05336. arXiv: 1510.05336 [stat.ML] (pg. 137).

[Ber07] Bergstrom, Carl. "Eigenfactor: Measuring the value and prestige of scholarly journals". In: *College & Research Libraries News* 68.5 (2007), pp. 314–316 (pg. 116, 117, 131).

[Ber18] Berger, Emery. *CSRankings: Computer Science Rankings*. http://csrankings.org/. 2018 (pg. 116–118, 123, 127, 131, 132, 134).

[BKÖ08] Bashirov, Agamirza E, Kurpınar, Emine Mısırlı, and Özyapıcı, Ali. "Multiplicative calculus and its applications". In: *Journal of Mathematical Analysis and Applications* 337.1 (2008), pp. 36–48. ISSN: 0022-247X. DOI: https://doi.org/10.1016/j.jmaa.2007.03.081. URL: http://www.

sciencedirect.com/science/article/pii/S0022247X07003824 (pg. 69).

[BKV09] Bell, Robert M., Koren, Yehuda, and Volinsky, Chris. *The BellKor solution to the Netflix Prize*. 2009 (pg. 128).

[Bla+18] Blackburn, Steve, Brodley, Carla, Jagadish, H. V., McKinley, Kathryn S, Nascimento, Mario, Shin, Minjeong, Stockwel, Sean, Xie, Lexing, and Xu, Qiongkai. *csmetrics.org: Institutional Publication Metrics for Computer Science*. https://github.com/csmetrics/csmetrics.org/blob/master/docs/Overview.md. 2018 (pg. 116, 131, 132).

[BLD20] Bangaru, Sai, Li, Tzu-Mao, and Durand, Frédo. "Unbiased Warped-Area Sampling for Differentiable Rendering". In: *ACM Trans. Graph.* 39.6 (2020), 245:1–245:18 (pg. 8).

[Bli07] Blinn, James F. "How to Solve a Cubic Equation, Part 5: Back to Numerics". In: *IEEE Computer Graphics and Applications* 27.3 (2007), pp. 78–89. DOI: 10.1109/MCG.2007.60 (pg. 12).

[Bli82] Blinn, James F. "A Generalization of Algebraic Surface Drawing". In: *ACM Trans. Graph.* 1.3 (July 1982), pp. 235–256. ISSN: 0730-0301. DOI: 10.1145/357306.357310 (pg. 5, 7, 9, 10, 23, 44, 46).

[BLK17] Bachem, Olivier, Lucic, Mario, and Krause, Andreas. *Scalable k-Means Clustering via Lightweight Coresets*. 2017. DOI: 10.48550/ARXIV.1702.08248. URL: https://arxiv.org/abs/1702.08248 (pg. 88).

[BNJ03] Blei, David M, Ng, Andrew Y, and Jordan, Michael I. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022 (pg. 136).

[BPF15] Brubaker, M, Punjani, A, and Fleet, D. "Building proteins in a day". In: *CVPR* (June 2015). DOI: 10.1109/cvpr.2015.7298929 (pg. 23, 153).

[Bra+18] Bradbury, James, Frostig, Roy, Hawkins, Peter, Johnson, Matthew James, Leary, Chris, Maclaurin, Dougal, Necula, George, Paszke, Adam, VanderPlas, Jake, Wanderman-Milne, Skye, and Zhang, Qiao. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018. URL: http://github.com/google/jax (pg. 7, 10, 15, 50, 61).

[Bra00] Bradski, G. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000) (pg. 94).

[But11] Butler, D. "Computing giants launch free science metrics". In: *Nature* 476.7358 (Aug. 2011), p. 18 (pg. 115).

[CAL15] Clauset, Aaron, Arbesman, Samuel, and Larremore, Daniel B. "Systematic inequality and hierarchy in faculty hiring networks". In: *Science Advances* (2015) (pg. 116, 118, 131, 132, 136).

[Car+21] Caron, Mathilde, Touvron, Hugo, Misra, Ishan, Jégou, Hervé, Mairal, Julien, Bojanowski, Piotr, and Joulin, Armand. "Emerging Proper-

ties in Self-Supervised Vision Transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 9650–9660. arXiv: 2104.14294 [cs.CV] (pg. 48).

[CAS15] Choudhury, Sanjiban, Arora, Sankalp, and Scherer, Sebastian. "The planner ensemble: Motion planning by executing diverse algorithms". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2389–2395. DOI: 10.1109/ICRA.2015.7139517 (pg. 87).

[Cat72] Catmull, Edwin. "A System for Computer Generated Movies". In: *Proceedings of the ACM Annual Conference - Volume 1*. ACM '72. Boston, Massachusetts, USA: Association for Computing Machinery, 1972, pp. 422–431. ISBN: 9781450374910. DOI: 10.1145/800193.569952 (pg. 2).

[CBK05] Cheung, Kong-man German, Baker, Simon, and Kanade, Takeo. "Shape-from-silhouette across time part i: Theory and algorithms". In: *International Journal of Computer Vision* 62.3 (2005), pp. 221–247 (pg. 20).

[CCS12] Corsini, Massimiliano, Cignoni, Paolo, and Scopigno, Roberto. "Efficient and flexible sampling with blue noise properties of triangular meshes". In: *TVCG* 18.6 (2012), pp. 914–924. DOI: 10.1109/TVCG.2012.34. URL: http://vcg.isti.cnr.it/Publications/2012/CCS12/TVCG-2011-07-0217.pdf (pg. 73, 82, 83).

[Che+19] Chen, Wenzheng, Ling, Huan, Gao, Jun, Smith, Edward, Lehtinen, Jaakko, Jacobson, Alec, and Fidler, Sanja. "Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer". In: *Advances in Neural Information Processing Systems* 32 (2019). Ed. by Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. arXiv: 1908.01210 [cs.CV] (pg. 9).

[Che+23] Chen, Zhiqin, Funkhouser, Thomas, Hedman, Peter, and Tagliasacchi, Andrea. "MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures". In: *The Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023 (pg. 47, 55).

[Chr+17] Christiano, Paul F, Leike, Jan, Brown, Tom, Martic, Miljan, Legg, Shane, and Amodei, Dario. "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. arXiv: 1706.03741 [stat.ML] (pg. 108).

[CN11] Coates, Adam and Ng, Andrew Y. "The importance of encoding versus training with sparse coding and vector quantization". In: *ICML*. 2011, pp. 921–928 (pg. 120).

[Col+10] Colledge, Lisa, Moya-Anegón, Félix de, Guerrero-Bote, Vicente P, López-Illescas, Carmen, Moed, Henk F, et al. "SJR and SNIP: two new journal metrics in Elsevier's Scopus". In: *Insights* 23.3 (2010), p. 215 (pg. 115).

[Col+14] Coleman, David, Sucan, Ioan, Chitta, Sachin, and Correll, Nikolaus. *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*. 2014. DOI: 10.48550/ARXIV.1404.3785. URL: https://arxiv.org/abs/1404.3785 (pg. 86).

[Col+21] Cole, Forrester, Genova, Kyle, Sud, Avneesh, Vlasic, Daniel, and Zhang, Zhoutong. *Differentiable Surface Rendering via Non-Differentiable Sampling*. 2021. arXiv: 2108.04886 [cs.GR] (pg. 8).

[Coo66] Coons, Steven Anson. "The Uses of Computers in Technology". In: *Scientific American* 215.3 (1966), pp. 176–191. URL: http://www.jstor.org/stable/24931054 (pg. 2, 3).

[CS22] Cheng, Ho Kei and Schwing, Alexander G. "XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model". In: *ECCV*. 2022 (pg. 50, 65).

[Dan59] Daniel, Cuthbert. "Use of Half-Normal Plots in Interpreting Factorial Two-Level Experiments". In: *Technometrics* 1.4 (1959), pp. 311–341. DOI: 10.1080/00401706.1959.10489866 (pg. 63).

[DLR77] Dempster, A.P., Laird, N.M., and Rubin, Donald B. "Maximum likelihood from incomplete data via the EM algorithm". In: *J. Royal Stat. Soc. (B)* 39.1 (1977), pp. 1–38. ISSN: 00359246. DOI: http://dx.doi.org/10.2307/2984875. arXiv: 0710.5696v2. URL: http://www.jstor.org/stable/10.2307/2984875 (pg. 14, 68, 70, 72).

[Doy60] Doyle, Worthie. "Recognition of Sloppy, Hand-Printed Characters". In: IRE-AIEE-ACM '60 (Western). San Francisco, California: Association for Computing Machinery, 1960, pp. 133–142. ISBN: 9781450378697. DOI: 10.1145/1460361.1460380 (pg. 2).

[DSM18] Dhawale, Aditya, Shaurya Shankar, Kumar, and Michael, Nathan. "Fast Monte-Carlo Localization on Aerial Vehicles Using Approximate Continuous Belief Representations". In: *CVPR*. 2018, pp. 5851–5859 (pg. 75).

[Dys09] Dyson, Freeman. "Birds and frogs". In: *Notices of the AMS* 56.2 (2009), pp. 212–223 (pg. 151).

[Eck+15] Eckart, Ben, Kim, Kihwan, Troccoli, Alejandro, Kelly, Alonzo, and Kautz, Jan. "MLMD: Maximum Likelihood Mixture Decoupling for Fast and Accurate Point Cloud Registration". In: *3DV*. 2015, pp. 241–249. ISBN: 9781467383325. DOI: 10.1109/3DV.2015.34. URL: http://jankautz.com/publications/MLMD%5C_3DV15.pdf (pg. 9, 46, 67).

[Eck+16] Eckart, Ben, Kim, Kihwan, Troccoli, Alejandro, Kelly, Alonzo, and Kautz, Jan. "Accelerated Generative Models for 3D Point Cloud Data".

In: *CVPR*. 2016, pp. 5497–5505. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.593. URL: http://ieeexplore.ieee.org/document/7780962/ (pg. 9, 14, 23, 46, 47, 67, 76).

[Edu18] Education, Times Higher. *World University Rankings, Computer Science*. https://www.timeshighereducation.com/world-university-rankings/2018/subject-ranking/computer-science. 2018 (pg. 116, 131, 132).

[Eim+21] Eimer, Theresa, Biedenkapp, André, Reimer, Maximilian, Adriaensen, Steven, Hutter, Frank, and Lindauer, Marius. "DACBench: A Benchmark Library for Dynamic Algorithm Configuration". In: (2021). DOI: 10.48550/ARXIV.2105.08541. URL: https://arxiv.org/abs/2105.08541 (pg. 88, 90).

[EKC16] Engel, Jakob, Koltun, Vladlen, and Cremers, Daniel. *Direct Sparse Odometry*. 2016. DOI: 10.48550/ARXIV.1607.02565. URL: https://arxiv.org/abs/1607.02565 (pg. 86, 93).

[EKK18] Eckart, B., Kim, K., and Kautz, J. "HGMR: Hierarchical Gaussian Mixtures for Adaptive 3D Registration". In: *ECCV*. Sept. 2018, pp. 730–746. ISBN: 978-3-030-01267-0. URL: http://arxiv.org/abs/1807.02587 (pg. 9, 14, 46, 47, 67, 75, 76, 84).

[End+10] Enderton, Eric, Sintorn, Erik, Shirley, Peter, and Luebke, David. "Stochastic Transparency". In: *I3D '10: Proceedings of the 2010 symposium on Interactive 3D graphics and games*. Washington, DC, USA, 2010, pp. 157–164. ISBN: 978-1-60558-938-1 (pg. 13, 48).

[Eva66] Evans, David C. *Graphical man/machine communications*. Tech. rep. University of Utah, 1966. URL: https://collections.lib.utah.edu/ark:/87278/s61n8j92 (pg. 3).

[Fal+08] Falagas, Matthew E., Kouranos, Vasilios D., Arencibia-Jorge, Ricardo, and Karageorgopoulos, Drosos E. "Comparison of SCImago journal rank indicator with journal impact factor". In: *The FASEB Journal* 22.8 (2008), pp. 2623–2628 (pg. 117, 118).

[Far03] Farnebäck, Gunnar. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Image Analysis*. Ed. by Bigun, Josef and Gustavsson, Tomas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370. ISBN: 978-3-540-45103-7 (pg. 53–55).

[Fei69] Feigenbaum, Edward A. "Artificial Intelligence: Themes in the Second Decade". In: *Information Processing 68*. Amsterdam: North-Holland Publishing Company, 1969. URL: https://apps.dtic.mil/sti/citations/AD0680487 (pg. 3).

[Fen13] Fenner, Martin. "What can article-level metrics do for you?" In: *PLoS biology* 11.10 (2013), e1001687 (pg. 115).

[Fey51] Feynman, Richard P. "An operator calculus having applications in quantum electrodynamics". In: *Physical Review* 84.1 (1951), p. 108 (pg. 69).

[Fou18] Foundation, National Science. *Download Awards by Year*. https://www.nsf.gov/awardsearch/download.jsp. 2018 (pg. 119).

[Fox01] Fox, Dieter. "KLD-Sampling: Adaptive Particle Filters". In: *Advances in Neural Information Processing Systems*. Ed. by Dietterich, T., Becker, S., and Ghahramani, Z. Vol. 14. MIT Press, 2001. URL: http://papers.neurips.cc/paper/1998-kld-sampling-adaptive-particle-filters.pdf (pg. 2).

[Fra04] Frahm, Gabriel. "Generalized Elliptical Distributions: Theory and Applications". PhD thesis. Universität zu Köln, 2004. URL: https://kups.ub.uni-koeln.de/1319/ (pg. 149).

[Fra18] Frazier, Peter I. *A Tutorial on Bayesian Optimization*. 2018. DOI: 10.48550/ARXIV.1807.02811. URL: https://arxiv.org/abs/1807.02811 (pg. 90).

[Fri+22] Fridovich-Keil, Sara, Yu, Alex, Tancik, Matthew, Chen, Qinhong, Recht, Benjamin, and Kanazawa, Angjoo. "Plenoxels: Radiance Fields without Neural Networks". In: *CVPR*. 2022 (pg. 51).

[FS97] Freund, Yoav and Schapire, Robert E. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1504. URL: http://www.sciencedirect.com/science/article/pii/S002200009791504X (pg. 128).

[FX89] Fang, Kai-Tai and Xu, Jian-Lun. *A Class of Multivariate Distributions Including the Multivariate Logistic*. 1989 (pg. 149).

[Gab15] Gaboury, Jacob. "Hidden Surface Problems: On the Digital Image as Material Object". In: *Journal of Visual Culture* 14.1 (2015), pp. 40–60. DOI: 10.1177/1470412914562270 (pg. 4).

[GAP08] Gupta, Ankit, Alliez, Pierre, and Pion, Sylvain. *Principal Component Analysis in CGAL*. Research Report RR-6642. INRIA, 2008, p. 13. URL: https://hal.inria.fr/inria-00327027 (pg. 72, 75).

[Gei+96] Geist, Robert, Chetuparambil, Madhu, Hedetniemi, Stephen, and Turner, A. Joe. "Computing Research Programs in the U.S." In: *Commun. ACM* 39.12 (Dec. 1996), pp. 96–99. ISSN: 0001-0782. DOI: 10.1145/240483.240505. URL: http://doi.acm.org/10.1145/240483.240505 (pg. 117).

[Gen+20] Genova, Kyle, Cole, Forrester, Sud, Avneesh, Sarna, Aaron, and Funkhouser, Thomas. *Local Deep Implicit Functions for 3D Shape*. 2020. arXiv: 1912.06126 [cs.CV] (pg. 9, 46).

[GG17] Galiani, Sebastian and Gálvez, Ramiro H. *The life cycle of scholarly articles across fields of research*. Tech. rep. National Bureau of Economic Research, 2017 (pg. 115, 117).

[GH97] Garland, Michael and Heckbert, Paul S. "Surface simplification using quadric error metrics". In: *SIGGRAPH*. 1997, pp. 209–216. ISBN: 0897918967. DOI: 10.1145/258734.258849. arXiv: 1708.07199. URL: http://portal.acm.org/citation.cfm?doid=258734.258849 (pg. 15, 47, 73, 78, 82, 83).

[GLU12] Geiger, Andreas, Lenz, Philip, and Urtasun, Raquel. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 (pg. 87, 94).

[GM04] Guerri, Alessio and Milano, Michela. "Learning Techniques for Automatic Algorithm Portfolio Selection". In: *Proceedings of the 16th European Conference on Artificial Intelligence*. ECAI'04. Valencia, Spain: IOS Press, 2004, pp. 475–479. ISBN: 9781586034528 (pg. 88).

[GMT23] Goel, Kshitij, Michael, Nathan, and Tabib, Wennie. "Probabilistic Point Cloud Modeling via Self-Organizing Gaussian Mixture Models". In: *IEEE Robotics and Automation Letters* 8.5 (2023), pp. 2526–2533. DOI: 10.1109/LRA.2023.3256923 (pg. 46).

[Gol+17] Golovin, Daniel, Solnik, Benjamin, Moitra, Subhodeep, Kochanski, Greg, Karro, John Elliot, and Sculley, D., eds. *Google Vizier: A Service for Black-Box Optimization*. 2017, pp. 1487–1495. URL: http://www.kdd.org/kdd2017/papers/view/google-vizier-a-service-for-black-box-optimization (pg. 90, 91).

[Gou+10] Gourmel, Olivier, Pajot, Anthony, Paulin, Mathias, Barthe, Loic, and Poulin, Pierre. "Fitted BVH for Fast Raytracing of Metaballs". In: *Computer Graphics Forum* 3 (2010), pp. 7–288. DOI: 10.1111/j.1467-8659.2009.01597.x. URL: https://hal.archives-ouvertes.fr/hal-01516266 (pg. 9, 46).

[Gru17] Grupp, Michael. *evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo. 2017 (pg. 96).

[GSB14] Gammell, Jonathan D., Srinivasa, Siddhartha S., and Barfoot, Timothy D. "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 2997–3004. DOI: 10.1109/IROS.2014.6942976 (pg. 95).

[Gue83] Guenther, Raymond A. "Product integrals and sum integrals". In: *International Journal of Mathematical Education in Science and Technology* 14.2 (1983), pp. 243–249 (pg. 69).

[Guo+23] Guo, Yuan-Chen, Cao, Yan-Pei, Wang, Chen, He, Yu, Shan, Ying, Qie, Xiaohu, and Zhang, Song-Hai. *VMesh: Hybrid Volume-Mesh Representation for Efficient View Synthesis*. 2023. arXiv: 2303.16184 [cs.CV] (pg. 47, 55).

[Guz67] Guzman-Arenas, Adolfo. "Some Aspects of Pattern Recognition by Computer". MS thesis. Massachusetts Institute of Technology, 1967. URL: http://dspace.mit.edu/handle/1721.1/6887 (pg. 2, 3).

[HAB19] Hansen, Nikolaus, Akimoto, Youhei, and Baudis, Petr. *CMA-ES/pycma on Github*. Zenodo, DOI:10.5281/zenodo.2559634. Feb. 2019. DOI: 10.5281/zenodo.2559634. URL: https://doi.org/10.5281/zenodo.2559634 (pg. 27, 33).

[Hag08] Hagen, Nils T. "Harmonic Allocation of Authorship Credit: Source-Level Correction of Bibliometric Bias Assures Accurate Publication and Citation Analysis". In: *PLOS ONE* 3.12 (Dec. 2008), pp. 1–7. DOI: 10.1371/journal.pone.0004021. URL: https://doi.org/10.1371/journal.pone.0004021 (pg. 128).

[Han+21] Hansen, Nikolaus, Auger, Anne, Ros, Raymond, Mersmann, Olaf, Tušar, Tea, and Brockhoff, Dimo. "COCO: a platform for comparing continuous optimizers in a black-box setting". In: *Optimization Methods and Software* 36.1 (2021), pp. 114–144. DOI: 10.1080/10556788.2020.1808977. eprint: https://doi.org/10.1080/10556788.2020.1808977. URL: https://doi.org/10.1080/10556788.2020.1808977 (pg. 88, 109).

[Han+23] Hansen, Nikolaus, yoshihikoueno, ARF1, Kadlecová, Gabriela, Nozawa, Kento, Rolshoven, Luca, Chan, Matthew, Akimoto, Youhei, brieglhostis, and Brockhoff, Dimo. *CMA-ES/pycma: r3.3.0*. Version r3.3.0. Jan. 2023 (pg. 110).

[Han16] Hansen, Nikolaus. *The CMA Evolution Strategy: A Tutorial*. 2016. arXiv: 1604.00772 [cs.LG] (pg. 27, 33, 87, 90, 108–110).

[Has66] Hasselblad, Victor. "Estimation of Parameters for a Mixture of Normal Distributions". In: *Technometrics* 8.3 (1966), pp. 431–444. ISSN: 15372723. DOI: 10.1080/00401706.1966.10490375. URL: http://www.tandfonline.com/action/journalInformation?journalCode=utch20 (pg. 68).

[He+17] He, Kaiming, Gkioxari, Georgia, Dollár, Piotr, and Girshick, Ross B. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988 (pg. 3, 22, 24, 25).

[Hec86] Heckbert, Paul S. "Fun With Gaussians". In: *SIGGRAPH '86 Advanced Image Processing seminar notes* (1986) (pg. 9, 44).

[Her+20] Hertz, Amir, Hanocka, Rana, Giryes, Raja, and Cohen-Or, Daniel. "PointGMM: a Neural GMM Network for Point Clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020. arXiv: 2003.13326 [cs.LG] (pg. 9, 46).

[HFF22] Harley, Adam W., Fang, Zhaoyuan, and Fragkiadaki, Katerina. "Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories". In: *ECCV*. 2022 (pg. 52).

[HGC81] Hodge, Susan E., Greenberg, David A., and Challice, C. E. "Publication Credit". In: *Science* 213.4511 (1981), pp. 950–952. ISSN: 00368075, 10959203. URL: http://www.jstor.org/stable/1687033 (pg. 128).

[HH18] Huangfu, Q. and Hall, J. A. J. "Parallelizing the dual revised simplex method". In: *Mathematical Programming Computation* 10.1 (Mar. 2018), pp. 119–142. ISSN: 1867-2957. DOI: 10.1007/s12532-017-0130-5. URL: https://doi.org/10.1007/s12532-017-0130-5 (pg. 90).

[Hir05] Hirsch, J E. "An index to quantify an individual's scientific research output". In: *Proc. Natl. Acad. Sci. U. S. A.* 102.46 (Nov. 2005), pp. 16569–16572. ISSN: 0027-8424. DOI: 10.1073/pnas.0507655102. URL: https://www.ncbi.nlm.nih.gov/pubmed/16275915%20https://www.ncbi.nlm.nih.gov/pmc/PMC1283832/ (pg. 116, 129, 133).

[Hir08] Hirschmuller, Heiko. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), pp. 328–341. DOI: 10.1109/TPAMI.2007.1166 (pg. 94, 110).

[HK09] Hartley, Richard I. and Kahl, Fredrik. "Global Optimization through Rotation Space Search". In: *IJCV* 82.1 (Apr. 2009), pp. 64–79. ISSN: 1573-1405. DOI: 10.1007/s11263-008-0186-9. URL: https://doi.org/10.1007/s11263-008-0186-9 (pg. 77).

[HK17a] Hu, Humphrey and Kantor, George. "Efficient Automatic Perception System Parameter Tuning On Site without Expert Supervision". In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by Levine, Sergey, Vanhoucke, Vincent, and Goldberg, Ken. Vol. 78. Proceedings of Machine Learning Research. PMLR, Nov. 2017, pp. 57–66. URL: https://proceedings.mlr.press/v78/hu17a.html (pg. 87, 96).

[HK17b] Hu, Humphrey and Kantor, George. "Introspective Evaluation of Perception Performance for Parameter Tuning without Ground Truth". In: *Proceedings of Robotics: Science and Systems*. Cambridge, Massachusetts, July 2017. DOI: 10.15607/RSS.2017.XIII.033 (pg. 87).

[HK18] Hu, Humphrey and Kantor, George. "Compensating for Context by Learning Local Models of Perception Performance". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 4629–4634. DOI: 10.1109/IROS.2018.8593778 (pg. 87).

[Hor19] Horvath, Robert. "Image-Space Metaballs Using Deep Learning". MA thesis. Faculty of Informatics, TU Wien, July 2019. URL: https://www.cg.tuwien.ac.at/research/publications/2019/horvath-2018-ism/ (pg. 9, 46).

[Hu+18] Hu, Yixin, Zhou, Qingnan, Gao, Xifeng, Jacobson, Alec, Zorin, Denis, and Panozzo, Daniele. "Tetrahedral Meshing in the Wild". In: *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201353 (pg. 8).

[Hu+22] Hu, Shi-Min, Liu, Zheng-Ning, Guo, Meng-Hao, Cai, Junxiong, Huang, Jiahui, Mu, Tai-Jiang, and Martin, Ralph R. "Subdivision-based Mesh Convolution Networks". In: *ACM Trans. Graph.* 41.3 (2022), 25:1–25:16. DOI: 10.1145/3506694. URL: https://doi.org/10.1145/3506694 (pg. 55).

[Hua+20] Huang, H., Ye, H., Sun, Y., and Liu, M. "GMMLoc: Structure Consistent Visual Localization With Gaussian Mixture Models". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5043–5050. DOI: 10.1109/LRA.2020.3005130 (pg. 9, 46).

[Hua18] Huang, Jeff. *Best Paper Awards in Computer Science (since 1996)*. https://jeffhuang.com/best_paper_awards.html. 2018 (pg. 128, 132).

[Hub64] Huber, Peter J. "Robust Estimation of a Location Parameter". In: *Ann. Math. Statist.* 35.1 (Mar. 1964), pp. 73–101. DOI: 10.1214/aoms/1177703732. URL: https://doi.org/10.1214/aoms/1177703732 (pg. 121).

[Hub85] Huber, Peter J. "Projection Pursuit". In: *The Annals of Statistics* 13.2 (1985), pp. 435–475. ISSN: 00905364. URL: http://www.jstor.org/stable/2241175 (pg. 120).

[Ian+16] Iandola, Forrest N., Han, Song, Moskewicz, Matthew W., Ashraf, Khalid, Dally, William J., and Keutzer, Kurt. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size*. 2016. DOI: 10.48550/ARXIV.1602.07360. URL: https://arxiv.org/abs/1602.07360 (pg. 94).

[ID18] Insafutdinov, Eldar and Dosovitskiy, Alexey. *Unsupervised Learning of Shape and Pose with Differentiable Point Clouds*. 2018. arXiv: 1810.09381 [cs.CV] (pg. 8, 46).

[Ilh+21] Ilharco, Gabriel et al. *OpenCLIP*. Version 0.1. July 2021. DOI: 10.5281/zenodo.5143773. URL: https://doi.org/10.5281/zenodo.5143773 (pg. 48).

[Ins18] Institute, Nevada Policy Research. *Transparent California*. https://transparentcalifornia.com/agencies/salaries/#university-system. 2018 (pg. 119).

[Joh18] Johnes, Jill. "University rankings: What do they really show?" In: *Scientometrics* 115.1 (Apr. 2018), pp. 585–606. ISSN: 1588-2861. DOI: 10.1007/s11192-018-2666-1. URL: https://doi.org/10.1007/s11192-018-2666-1 (pg. 131).

[Jor09] Jordan, Michael I. "The Multivariate Gaussian". In: (2009). https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter13.pdf (pg. 70).

[JUD00] Julier, S., Uhlmann, J., and Durrant-Whyte, H.F. "A new method for the nonlinear transformation of means and covariances in filters and estimators". In: *IEEE Transactions on Automatic Control* 45.3 (2000), pp. 477–482. DOI: 10.1109/9.847726 (pg. 47).

[Jun+22] Jun-Seong, Kim, Yu-Ji, Kim, Ye-Bin, Moon, and Oh, Tae-Hyun. "HDR-Plenoxels: Self-Calibrating High Dynamic Range Radiance Fields". In: *ECCV*. 2022 (pg. 52).

[JV11] Jian, B. and Vemuri, B. C. "Robust Point Set Registration Using Gaussian Mixture Models". In: *PAMI* 33.8 (2011), pp. 1633–1645. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.223 (pg. 67, 76).

[Kad+10] Kadioglu, Serdar, Malitsky, Yuri, Sellmann, Meinolf, and Tierney, Kevin. "ISAC –Instance-Specific Algorithm Configuration". In: *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. NLD: IOS Press, 2010, pp. 751–756. ISBN: 9781607506058 (pg. 88).

[KB15] Kingma, Diederik P. and Ba, Jimmy. "Adam: A Method for Stochastic Optimization". In: *ICLR (Poster)*. 2015. URL: http://arxiv.org/abs/1412.6980 (pg. 20, 51).

[KBH06] Kazhdan, Michael, Bolitho, Matthew, and Hoppe, Hugues. "Poisson Surface Reconstruction". In: *Symposium on Geometry Processing*. Ed. by Sheffer, Alla and Polthier, Konrad. The Eurographics Association, 2006. ISBN: 3-905673-24-X. DOI: 10.2312/SGP/SGP06/061-070 (pg. 14, 36, 55, 56, 60, 66).

[KCD09] Kobilarov, Marin, Crane, Keenan, and Desbrun, Mathieu. "Lie group integrators for animation and control of vehicles". In: *ACM Trans. Graph.* 28 (2 May 2009) (pg. 38).

[Ker+23] Kerbl, Bernhard, Kopanas, Georgios, Leimkühler, Thomas, and Drettakis, George. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *ACM Transactions on Graphics (ToG)* 42.4 (July 2023) (pg. 43–47, 49, 51, 52, 55, 57, 58, 60, 61, 63).

[Kes+17] Keselman, Leonid, Woodfill, John Iselin, Grunnet-Jepsen, Anders, and Bhowmik, Achintya. "Intel RealSense Stereoscopic Depth Cameras". In: *CVPR Workshops* (2017). DOI: 10.1109/CVPRW.2017.167. arXiv: 1705.05548 [cs.CV] (pg. 3, 22, 75, 86, 87, 93, 96, 110, 113).

[Kes+23] Keselman, Leonid, Shih, Katherine, Hebert, Martial, and Steinfeld, Aaron. "Optimizing Algorithms From Pairwise User Preferences". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2023. arXiv: 2308.04571 [cs.RO] (pg. 1, 108).

[Kes19] Keselman, Leonid. "Venue Analytics: A Simple Alternative to Citation-Based Metrics". In: *ACM/IEEE Joint Conference on Digital Libraries*. 2019. DOI: 10.1109/JCDL.2019.00052. arXiv: 1904.12573 [cs.DL] (pg. 1, 114).

[KH13] Kazhdan, Michael and Hoppe, Hugues. "Screened poisson surface reconstruction". In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), pp. 1–13 (pg. 55, 57, 60, 64).

[KH17] Kurtz, Michael J. and Henneken, Edwin A. "Measuring metrics - a 40-year longitudinal cross-validation of citations, downloads, and peer review in astrophysics". In: *JAIST* 68.3 (2017), pp. 695–708. DOI: 10.1002/asi.23689. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.23689 (pg. 117).

[KH19] Keselman, Leonid and Hebert, Martial. "Direct Fitting of Gaussian Mixture Models". In: *2019 16th Conference on Computer and Robot Vision (CRV)*. 2019, pp. 25–32. DOI: 10.1109/CRV.2019.00012. arXiv: 1904.05537 [cs.CV] (pg. 1, 14, 66).

[KH22] Keselman, Leonid and Hebert, Martial. "Approximate Differentiable Rendering with Algebraic Surfaces". In: *European Conference on Computer Vision (ECCV)*. Oct. 2022. DOI: 10.1007/978-3-031-19824-3_35. arXiv: 2207.10606 [cs.CV] (pg. 1, 5, 44–48, 50–52, 55, 57, 58, 60, 61, 63, 93, 94).

[KH23a] Keselman, Leonid and Hebert, Martial. "Discovering Multiple Algorithm Configurations". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023. DOI: 10.1109/ICRA48891.2023.10160363. arXiv: 2303.07434 [cs.AI] (pg. 1, 86, 110).

[KH23b] Keselman, Leonid and Hebert, Martial. *Flexible Techniques for Differentiable Rendering with 3D Gaussians*. 2023. DOI: 10.48550/arXiv.2308.14737. arXiv: 2308.14737 [cs.CV] (pg. 1, 43).

[Kha19] Khallaghi, Siavash. *Pure Numpy Implementation of the Coherent Point Drift Algorithm*. https://github.com/siavashk/pycpd. 2019 (pg. 77, 84).

[Kin18] King, Davis. *Automatic learning rate scheduling that really works*. Feb. 2018. URL: http://blog.dlib.net/2018/02/automatic-learning-rate-scheduling-that.html (pg. 33).

[Kir+23] Kirillov, Alexander, Mintun, Eric, Ravi, Nikhila, Mao, Hanzi, Rolland, Chloe, Gustafson, Laura, Xiao, Tete, Whitehead, Spencer, Berg, Alexander C., Lo, Wan-Yen, Dollár, Piotr, and Girshick, Ross. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV] (pg. 65).

[Kir+57] Kirsch, Russell. A., Cahn, Lee, Ray, C., and Urban, Genevie. H. "Experiments in Processing Pictorial Information with a Digital Computer". In: IRE-ACM-AIEE '57 (Eastern). Washington, D.C.: Association for Computing Machinery, 1957, pp. 221–229. DOI: 10.1145/1457720.1457763 (pg. 3).

[Kon10] Konolige, Kurt. "Projected texture stereo". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 148–155. DOI: 10.1109/ROBOT.2010.5509796 (pg. 113).

[Kot+19] Kotthoff, Lars, Thornton, Chris, Hoos, Holger H., Hutter, Frank, and Leyton-Brown, Kevin. "Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA". In: *Automated Machine Learning: Methods, Systems, Challenges*. Ed. by Hutter, Frank, Kotthoff, Lars, and Vanschoren, Joaquin. Cham: Springer International Publishing, 2019, pp. 81–95. ISBN: 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5_4. URL: https://doi.org/10.1007/978-3-030-05318-5_4 (pg. 88).

[KRN97] Kanade, Takeo, Rander, Peter, and Narayanan, P.J. "Virtualized reality: constructing virtual worlds from real scenes". In: *IEEE MultiMedia* 4.1 (1997), pp. 34–47. DOI: 10.1109/93.580394 (pg. 52).

[Kro+18] Krotkov, Eric, Hackett, Douglas, Jackel, Larry, Perschbacher, Michael, Pippine, James, Strauss, Jesse, Pratt, Gill, and Orlowski, Christopher. "The DARPA robotics challenge finals: Results and perspectives". In: *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Springer, 2018, pp. 1–26 (pg. 87).

[KT01] Kaasalainen, M and Torppa, J. "Optimization methods for asteroid lightcurve inversion". In: *Icarus* 153.1 (2001), pp. 24–36 (pg. 23, 152).

[KUH17] Kato, Hiroharu, Ushiku, Yoshitaka, and Harada, Tatsuya. *Neural 3D Mesh Renderer*. 2017. arXiv: 1711.07566 [cs.CV] (pg. 8, 46).

[Lai+20] Laine, Samuli, Hellsten, Janne, Karras, Tero, Seol, Yeongho, Lehtinen, Jaakko, and Aila, Timo. "Modular Primitives for High-Performance

Differentiable Rendering". In: *ACM Transactions on Graphics* 39.6 (2020) (pg. 6, 8, 20, 23, 46).

[Lai21] Lai, Tin. "sbp-env: A Python Package for Sampling-based Motion Planner and Samplers". In: *Journal of Open Source Software* 6.66 (2021), p. 3782. DOI: 10.21105/joss.03782. URL: https://doi.org/10.21105/joss.03782 (pg. 95).

[Lav98] Lavalle, Steven M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. 1998 (pg. 93).

[LB14] Loper, Matthew M. and Black, Michael J. "OpenDR: An Approximate Differentiable Renderer". In: *ECCV*. Ed. by Fleet, David, Pajdla, Tomas, Schiele, Bernt, and Tuytelaars, Tinne. Cham: Springer International Publishing, 2014, pp. 154–169. ISBN: 978-3-319-10584-0 (pg. 8, 46).

[LB81] Lowe, David G. and Binford, Thomas O. "Interpretation Of Geometric Structure From Image Boundaries". In: *Techniques and Applications of Image Understanding*. Ed. by Pearson, James J. Vol. 0281. International Society for Optics and Photonics. SPIE, 1981, pp. 224–231. DOI: 10.1117/12.965752. URL: https://doi.org/10.1117/12.965752 (pg. 52).

[LBH15] LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539 (pg. 2).

[LC87] Lorensen, William E and Cline, Harvey E. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169 (pg. 14, 35, 55, 64).

[Lev+05] Levoy, Marc, Gerth, J, Curless, Brian, and Pull, K. *The Stanford 3D Scanning Repository*. 2005. URL: http://graphics.stanford.edu/data/3Dscanrep/ (pg. 14, 78).

[Ley+03] Leyton-Brown, Kevin, Nudelman, Eugene, Andrew, Galen, McFadden, Jim, and Shoham, Yoav. "A Portfolio Approach to Algorithm Select". In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. IJCAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc., 2003, pp. 1542–1543 (pg. 88).

[Ley02] Ley, Michael. "The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives". In: *String Processing and Information Retrieval*. Ed. by Laender, Alberto H. F. and Oliveira, Arlindo L. Springer Berlin Heidelberg, 2002, pp. 1–10. ISBN: 978-3-540-45735-0 (pg. 116, 118).

[LH16] Loshchilov, Ilya and Hutter, Frank. *CMA-ES for Hyperparameter Optimization of Deep Neural Networks*. 2016. arXiv: 1604.07269 [cs.NE] (pg. 27, 33, 87, 88).

[Li+20] Li, Lei, Zhu, Siyu, Fu, Hongbo, Tan, Ping, and Tai, Chiew-Lan. *End-to-End Learning Local Multi-view Descriptors for 3D Point Clouds*. 2020. arXiv: 2003.05855 [cs.CV] (pg. 8).

[Li+21] Li, Zhengqi, Niklaus, Simon, Snavely, Noah, and Wang, Oliver. "Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (pg. 52).

[Liu+19] Liu, Shichen, Li, Tianye, Chen, Weikai, and Li, Hao. "Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019. arXiv: 1904.01786 [cs.CV] (pg. 8, 15–18, 21, 23, 27, 32, 44, 46, 55).

[LJL06] Lehmann, Sune, Jackson, Andrew D, and Lautrup, Benny E. "Measures for measures". In: *Nature* 444.7122 (2006), p. 1003 (pg. 116).

[LKL18] Lin, Chen-Hsuan, Kong, Chen, and Lucey, Simon. "Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction". In: *AAAI Conf. on Artificial Intelligence (AAAI)*. 2018. arXiv: 1706.07036 [cs.CV] (pg. 8).

[LMS18] Lubienski, Sarah Theule, Miller, Emily K., and Saclarides, Evthokia Stephanie. "Sex Differences in Doctoral Student Publication Rates". In: *Educational Researcher* 47.1 (2018), pp. 76–81. DOI: 10.3102/0013189X17738746. URL: https://doi.org/10.3102/0013189X17738746 (pg. 120).

[LNS09] Leyton-Brown, Kevin, Nudelman, Eugene, and Shoham, Yoav. "Empirical Hardness Models: Methodology and a Case Study on Combinatorial Auctions". In: *J. ACM* 56.4 (July 2009). ISSN: 0004-5411. DOI: 10.1145/1538902.1538906. URL: https://doi.org/10.1145/1538902.1538906 (pg. 88).

[Lom+19] Lombardi, Stephen, Simon, Tomas, Saragih, Jason, Schwartz, Gabriel, Lehrmann, Andreas, and Sheikh, Yaser. "Neural Volumes: Learning Dynamic Renderable Volumes from Images". In: *ACM Trans. Graph.* 38.4 (July 2019), 65:1–65:14. arXiv: 1906.07751 (pg. 8).

[LRF19] Lai, Tin, Ramos, Fabio, and Francis, Gilad. "Balancing Global Exploration and Local-connectivity Exploitation with Rapidly-exploring Random disjointed-Trees". In: *Proceedings of The International Conference on Robotics and Automation (ICRA)*. IEEE. 2019 (pg. 95).

[Lui+23] Luiten, Jonathon, Kopanas, Georgios, Leibe, Bastian, and Ramanan, Deva. "Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis". In: (2023). arXiv: 2308.09713 [cs.CV] (pg. 46).

[Lv+18] Lv, Xueying, Wang, Yitian, Deng, Junyi, Zhang, Guanyu, and Zhang, Liu. "Improved Particle Swarm Optimization Algorithm Based on Last-Eliminated Principle and Enhanced Information Sharing". In: *Computational Intelligence and Neuroscience* 2018 (Dec. 2018), p. 5025672. ISSN: 1687-5265. DOI: 10.1155/2018/5025672. URL: https://doi.org/10.1155/2018/5025672 (pg. 93).

[LZ21] Lassner, Christoph and Zollhöfer, Michael. "Pulsar: Efficient Sphere-based Neural Rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 1440–1449. arXiv: 2004.07484 [cs.GR] (pg. 8, 15, 16, 18, 20, 23, 32, 46).

[MA+73] Malik, Henrick J, Abraham, Bovas, et al. "Multivariate logistic distributions". In: *The Annals of Statistics* 1.3 (1973), pp. 588–590 (pg. 149).

[MA83] Martin, Worthy N. and Aggarwal, J. K. "Volumetric Descriptions of Objects from Multiple Views". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5.2 (1983), pp. 150–158. DOI: 10.1109/TPAMI.1983.4767367 (pg. 21).

[Mac+17] MacGlashan, James, Ho, Mark K., Loftin, Robert, Peng, Bei, Wang, Guan, Roberts, David L., Taylor, Matthew E., and Littman, Michael L. "Interactive Learning from Policy-Dependent Human Feedback". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Precup, Doina and Teh, Yee Whye. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 2285–2294. arXiv: 1701.06049 [cs.AI] (pg. 108).

[Mag+09] Magnusson, M., Nuchter, A., Lorken, C., Lilienthal, A. J., and Hertzberg, J. "Evaluation of 3D registration reliability and speed - A comparison of ICP and NDT". In: *ICRA*. May 2009, pp. 3907–3912. DOI: 10.1109/ROBOT.2009.5152538 (pg. 76).

[Mag09] Magnusson, Martin. "The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection". PhD thesis. Örebro universitet, 2009. ISBN: 978-91-7668-696-6. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:oru:diva-8458 (pg. 46, 76).

[Mah36] Mahalanobis, Prasanta Chandra. "On the generalized distance in statistics". In: *Proceedings of the National Institute of Sciences (Calcutta)* (1936), pp. 49–55 (pg. 10, 47).

[Mak+21] Makatura, Liane, Guo, Minghao, Schulz, Adriana, Solomon, Justin, and Matusik, Wojciech. "Pareto Gamuts: Exploring Optimal Designs Across Varying Contexts". In: *ACM Transactions on Graphics (SIGGRAPH)*

40.4 (Aug. 2021), pp. 1–17. DOI: 10.1145/3450626.3459750. URL: https://doi.org/10.1145/3450626.3459750 (pg. 87).

[Man06] Man, Petr. "Generating and Real-Time Rendering of Clouds". In: *Central European seminar on computer graphics* (2006) (pg. 44).

[Mát65] Mátyáš, I. "Random Optimization". In: *Avtomat. i Telemekh* 26.2 (1965), pp. 246–253. URL: http://mi.mathnet.ru/at11288 (pg. 108, 112).

[Max95] Max, Nelson. "Optical models for direct volume rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108. DOI: 10.1109/2945.468400 (pg. 9).

[MB13] McGuire, Morgan and Bavoil, Louis. "Weighted Blended Order-Independent Transparency". In: *Journal of Computer Graphics Techniques (JCGT)* 2.2 (Dec. 2013), pp. 122–141. ISSN: 2331-7418. URL: http://jcgt.org/published/0002/02/09/ (pg. 13, 48).

[MBL19] Mirzasoleiman, Baharan, Bilmes, Jeff, and Leskovec, Jure. "Coresets for Data-efficient Training of Machine Learning Models". In: (2019). DOI: 10.48550/ARXIV.1906.01827. URL: https://arxiv.org/abs/1906.01827 (pg. 88).

[McD09] McDonald, John H. *Handbook of biological statistics*. Vol. 2. 2009 (pg. 149).

[MG15] Menze, Moritz and Geiger, Andreas. "Object Scene Flow for Autonomous Vehicles". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 (pg. 87).

[MG17a] Maystre, Lucas and Grossglauser, Matthias. "Just Sort It! A Simple and Effective Approach to Active Preference Learning". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 2344–2353 (pg. 109).

[MG17b] Ménard, Pierre and Garivier, Aurélien. "A minimax and asymptotically optimal algorithm for stochastic bandits". In: *Algorithmic Learning Theory*. 2017 Algorithmic Learning Theory Conference 76 (2017). URL: https://hal.archives-ouvertes.fr/hal-01475078 (pg. 97).

[MH08] Maaten, Laurens van der and Hinton, Geoffrey. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605 (pg. 136).

[Mik+13] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. "Efficient estimation of word representations in vector space". In: *ICLR* (2013) (pg. 120).

[Mil+20a] Mildenhall, Ben, Srinivasan, Pratul P., Tancik, Matthew, Barron, Jonathan T., Ramamoorthi, Ravi, and Ng, Ren. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *Computer Vision – ECCV 2020*. Ed. by Vedaldi, Andrea, Bischof, Horst, Brox, Thomas, and Frahm, Jan-Michael. Cham: Springer International Publishing, 2020,

pp. 405–421. ISBN: 978-3-030-58452-8. DOI: 10.1007/978-3-030-58452-8_24. arXiv: 2003.08934 (pg. 2, 6–9, 20, 21, 23, 43, 44, 49, 52, 55).

[Mil+20b] Miller, Ian D. et al. *Mine Tunnel Exploration using Multiple Quadrupedal Robots*. 2020. arXiv: 1909.09662 [cs.RO] (pg. 7).

[Min66] Minsky, Marvin L. "Artificial Intelligence". In: *Scientific American* 215.3 (1966), pp. 246–263. URL: http://www.jstor.org/stable/24931058 (pg. 3).

[Mir96] Mirtich, Brian. "Fast and Accurate Computation of Polyhedral Mass Properties". In: *JGT* 1.2 (1996), pp. 31–50. DOI: 10.1080/10867651.1996.10487458. eprint: https://doi.org/10.1080/10867651.1996.10487458. URL: https://doi.org/10.1080/10867651.1996.10487458 (pg. 47, 75).

[MO90] Marshall, Albert W. and Olkin, Ingram. "Multivariate Distributions Generated from Mixtures of Convolution and Product Families". In: *Lecture Notes-Monograph Series* 16 (1990), pp. 371–393. ISSN: 07492170. URL: http://www.jstor.org/stable/4355607 (pg. 149).

[Mor+18] Morgan, Allison C., Economou, Dimitrios J., Way, Samuel F., and Clauset, Aaron. "Prestige drives epistemic inequality in the diffusion of scientific ideas". In: *EPJ Data Science* 7.1 (Oct. 2018), p. 40. ISSN: 2193-1127 (pg. 120).

[Mor83] Moravec, Hans P. "The Stanford Cart and the CMU Rover". In: *Proceedings of the IEEE* 71.7 (1983), pp. 872–884. DOI: 10.1109/PROC.1983.12684 (pg. 2).

[MS09] Malitsky, Yuri and Sellmann, Meinolf. "Stochastic Offline Programming". In: *2009 21st IEEE International Conference on Tools with Artificial Intelligence*. 2009, pp. 784–791. DOI: 10.1109/ICTAI.2009.23 (pg. 88).

[MS10] Myronenko, A. and Song, X. "Point Set Registration: Coherent Point Drift". In: *PAMI* 32.12 (Dec. 2010), pp. 2262–2275. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.46 (pg. 76, 77, 84).

[Mül+22] Müller, Thomas, Evans, Alex, Schied, Christoph, and Keller, Alexander. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127 (pg. 44).

[Mun68] Munson, John H. "Experiments in the Recognition of Hand-Printed Text, Part I: Character Recognition". In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part II*. AFIPS '68 (Fall, part II). San Francisco, California: Association for Computing Machinery, 1968, pp. 1125–1138. ISBN: 9781450379007. DOI: 10.1145/1476706.1476735 (pg. 3).

[Mur91] Muraki, Shigeru. "Volumetric Shape Description of Range Data Using "Blobby Model"". In: *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '91. New York, NY, USA: Association for Computing Machinery, 1991, pp. 227–235. ISBN: 0897914368. DOI: 10.1145/122718.122743 (pg. 9, 46).

[MWC18] Morgan, Allison C., Way, Samuel F., and Clauset, Aaron. "Automatically assembling a full census of an academic field". In: *PLOS ONE* 13.8 (Aug. 2018), pp. 1–18. DOI: 10.1371/journal.pone.0202223. URL: https://doi.org/10.1371/journal.pone.0202223 (pg. 119).

[MWK22] Müller, Jan U., Weinmann, Michael, and Klein, Reinhard. "Unbiased Gradient Estimation for Differentiable Surface Splatting via Poisson Sampling". In: *European Conference on Computer Vision (ECCV)*. 2022 (pg. 9, 46).

[Nag68] Nagy, George. "State of the art in pattern recognition". In: *Proceedings of the IEEE* 56.5 (1968), pp. 836–863. DOI: 10.1109/PROC.1968.6414 (pg. 3).

[New+11] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. "KinectFusion: Real-time dense surface mapping and tracking". In: *ISMAR*. Oct. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378 (pg. 66).

[Nil65] Nilsson, Nils J. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*. Thanks to CMU Libraries for lending me Herbert Simon's personal copy. McGraw Hill, 1965 (pg. 3).

[Nim+19] Nimier-David, Merlin, Vicini, Delio, Zeltner, Tizian, and Jakob, Wenzel. "Mitsuba 2: A Retargetable Forward and Inverse Renderer". In: *ACM Trans. Graph.* 38.6 (Nov. 2019). ISSN: 0730-0301. DOI: 10.1145/3355089.3356498 (pg. 6, 46).

[Nis84] Nishihara, H.K. *PRISM: A Practical Real-Time Imaging Stereo Matcher*. Tech. rep. MIT AI Lab, May 1984 (pg. 110, 113).

[NM01] Nulkar, Manjushree and Mueller, Klaus. "Splatting With Shadows". In: *Volume Graphics 2001*. Ed. by Mueller, Klaus and Kaufman, Arie E. Vienna: Springer Vienna, 2001, pp. 35–49. ISBN: 978-3-7091-6756-4 (pg. 44).

[Noc+14] Nocka, Andrew, Zheng, Danning, Hu, Tianran, and Luo, Jiebo. "Moneyball for Academia: Toward Measuring and Maximizing Faculty Performance and Impact". In: *2014 IEEE International Conference on Data Mining Workshop*. Dec. 2014, pp. 193–197 (pg. 119).

[NR18] News, US and Report, World. *Best Computer Science Schools*. https://www.usnews.com/best-graduate-schools/top-science-schools/computer-science-rankings. 2018 (pg. 116, 131, 132).

[NW06] Nocedal, Jorge and Wright, Stephen J. *Numerical Optimization*. second. New York, NY, USA: Springer, 2006 (pg. 77, 79).

[Ope+19] OpenAI et al. *Learning Dexterous In-Hand Manipulation*. 2019. arXiv: 1808.00177 [cs.LG] (pg. 86).

[Oqu+23] Oquab, Maxime et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2023. arXiv: 2304.07193 [cs.CV] (pg. 48).

[OTM19] O'Meadhra, C., Tabib, W., and Michael, N. "Variable Resolution Occupancy Mapping Using Gaussian Mixture Models". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2015–2022. DOI: 10.1109/LRA.2018.2889348 (pg. 9, 46).

[Pag+99] Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. *The PageRank citation ranking: Bringing order to the web*. Tech. rep. Stanford, 1999 (pg. 118, 131).

[Pap66] Papert, Seymour A. "The Summer Vision Project". In: *Artificial Intelligence Group Memo* 100 (1966). URL: http://dspace.mit.edu/handle/1721.1/6125 (pg. 2).

[Par+19] Park, Jeong Joon, Florence, Peter, Straub, Julian, Newcombe, Richard, and Lovegrove, Steven. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (pg. 2).

[Pas+19] Paszke, Adam et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (pg. 94).

[Pau92] Paul J. Besl, Neil D. McKay. "Method for registration of 3-D shapes". In: *Proc.SPIE*. 1992, pp. 1611–1621. DOI: 10.1117/12.57955. URL: https://doi.org/10.1117/12.57955 (pg. 66, 77, 84, 85).

[Pea94] Pearson, Karl. "Contributions to the Mathematical Theory of Evolution". In: *Philosophical Transactions of the Royal Society of London. A* 185 (1894), pp. 71–110. ISSN: 02643820. URL: http://www.jstor.org/stable/90667 (pg. 44).

[Ped+11] Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python". In: *JMLR* 12 (2011), pp. 2825–2830 (pg. 74, 121).

[Pfi+00] Pfister, Hanspeter, Zwicker, Matthias, Baar, Jeroen van, and Gross, Markus. "Surfels: Surface Elements as Rendering Primitives". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 335–342. ISBN: 1581132085. DOI: 10.1145/344779.344936 (pg. 8, 46, 147).

[PM00] Pelleg, Dau and Moore, Andrew. "X-means: Extending K-means with Efficient Estimation of the Number of Clusters". In: *ICML*. 2000, pp. 727–734 (pg. 75).

[Pom88] Pomerleau, Dean A. "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *Advances in Neural Information Processing Systems*. Ed. by Touretzky, D. Vol. 1. Morgan-Kaufmann, 1988. URL: https://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network (pg. 2).

[PR69] Polak, E. and Ribiere, G. "Note sur la convergence de méthodes de directions conjuguées". In: *R.I.R.O.* 3.16 (1969), pp. 35–43. DOI: 10.1051/m2an/196903R100351. URL: https://doi.org/10.1051/m2an/196903R100351 (pg. 77).

[PSM14] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher. "Glove: Global vectors for word representation". In: *EMNLP*. 2014, pp. 1532–1543 (pg. 120).

[Rad+21] Radford, Alec, Kim, Jong Wook, Hallacy, Chris, Ramesh, A., Goh, Gabriel, Agarwal, Sandhini, Sastry, Girish, Askell, Amanda, Mishkin, Pamela, Clark, Jack, Krueger, Gretchen, and Sutskever, Ilya. "Learning Transferable Visual Models From Natural Language Supervision". In: *ICML*. 2021. arXiv: 2103.00020 [cs.CV] (pg. 48).

[Rak+23] Rakotosaona, Marie-Julie, Manhardt, Fabian, Arroyo, Diego Martin, Niemeyer, Michael, Kundu, Abhijit, and Tombari, Federico. *NeRFMeshing: Distilling Neural Radiance Fields into Geometrically-Accurate 3D Meshes*. 2023. arXiv: 2303.09431 [cs.CV] (pg. 47).

[Ran15] Rankings, Shanghai. *Academic Ranking of World Universities in Computer Science*. http://www.shanghairanking.com/SubjectCS2015.html. 2015 (pg. 131, 132).

[Ran18] Rankings, QS World University. *Computer Science & Information Systems*. https://www.topuniversities.com/university-rankings/university-subject-rankings/2018/computer-science-information-systems. 2018 (pg. 116, 131, 132).

[Rav+20] Ravi, Nikhila, Reizenstein, Jeremy, Novotny, David, Gordon, Taylor, Lo, Wan-Yen, Johnson, Justin, and Gkioxari, Georgia. "Accelerating 3D

Deep Learning with PyTorch3D". In: *arXiv:2007.08501* (2020) (pg. 8, 15, 16, 18–21, 32, 55).

[RB93] Rossignac, Jarek and Borrel, Paul. "Multi-resolution 3D approximations for rendering complex scenes". In: *Model. Comput. Graph.* Springer Verlag, 1993, pp. 455–465. ISBN: 978-3-642-78116-2. DOI: 10.1007/978-3-642-78114-8_29. URL: http://www.springerlink.com/index/10.1007/978-3-642-78114-8%5C_29 (pg. 73, 82, 83).

[RBK98] Rowley, H.A., Baluja, S., and Kanade, T. "Neural network-based face detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.1 (1998), pp. 23–38. DOI: 10.1109/34.655647 (pg. 2).

[Rei+21] Reizenstein, Jeremy, Shapovalov, Roman, Henzler, Philipp, Sbordone, Luca, Labatut, Patrick, and Novotny, David. "Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction". In: *International Conference on Computer Vision*. 2021 (pg. 22, 50, 52–55).

[RHL02] Rusinkiewicz, Szymon, Hall-Holt, Olaf, and Levoy, Marc. "Real-Time 3D Model Acquisition". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 438–446. ISSN: 0730-0301. DOI: 10.1145/566654.566600 (pg. 2).

[RHW86] Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687 (pg. 120).

[Ric76] Rice, John R. "The Algorithm Selection Problem". In: *Adv. Comput.* 15 (1976), pp. 65–118 (pg. 87, 88).

[RL01] Rusinkiewicz, S. and Levoy, M. "Efficient variants of the ICP algorithm". In: *Proceedings Third International Conference on 3D Digital Imaging & Modeling*. 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423 (pg. 17, 19).

[RM51] Robbins, Herbert and Monro, Sutton. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407 (pg. 121).

[Rob63] Roberts, Lawrence Gillman. "Machine perception of three-dimensional solids". PhD thesis. Massachusetts Institute of Technology, 1963. URL: http://dspace.mit.edu/handle/1721.1/11589 (pg. 2, 3).

[Ros60] Rosenblatt, Frank. "Perceptron Simulation Experiments". In: *Proceedings of the IRE* 48.3 (1960), pp. 301–309. DOI: 10.1109/JRPROC.1960.287598 (pg. 2).

[Rou87] Rousseeuw, Peter J. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: http://

www.sciencedirect.com/science/article/pii/0377042787901257 (pg. 137).

[RT07] Ren, Jie and Taylor, Richard N. "Automatic and versatile publications ranking for research institutions and scholars". In: *Communications of the ACM* 50.6 (2007), pp. 81–85 (pg. 117).

[RT18] Rapin, J. and Teytaud, O. *Nevergrad - A gradient-free optimization platform*. https://GitHub.com/FacebookResearch/Nevergrad. 2018 (pg. 90).

[Rus+17] Russo, Daniel, Van Roy, Benjamin, Kazerouni, Abbas, Osband, Ian, and Wen, Zheng. "A Tutorial on Thompson Sampling". In: (2017). DOI: 10.48550/ARXIV.1707.02038. URL: https://arxiv.org/abs/1707.02038 (pg. 91).

[Sam59] Samuel, Aruthur L. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210 (pg. 3).

[SC22] Stumberg, L. von and Cremers, D. "DM-VIO: Delayed Marginalization Visual-Inertial Odometry". In: *International Conference on Robotics and Automation (ICRA)* 7.2 (2022), pp. 1408–1415. DOI: 10.1109/LRA.2021.3140129 (pg. 96).

[Sch+14] Scharstein, Daniel, Hirschmüller, Heiko, Kitajima, York, Krathwohl, Greg, Nešić, Nera, Wang, Xi, and Westling, Porter. "High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth". In: *Pattern Recognition*. 2014 (pg. 87, 94, 110, 111).

[Sch+16] Schönberger, Johannes Lutz, Zheng, Enliang, Pollefeys, Marc, and Frahm, Jan-Michael. "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *European Conference on Computer Vision (ECCV)*. 2016 (pg. 24, 25, 43, 45, 50, 152).

[Sch+18] Schubert, D., Goll, T., Demmel, N., Usenko, V., Stueckler, J., and Cremers, D. "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry". In: *International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018 (pg. 96).

[Sch+22] Schede, Elias, Brandt, Jasmin, Tornede, Alexander, Wever, Marcel, Bengs, Viktor, Hüllermeier, Eyke, and Tierney, Kevin. *A Survey of Methods for Automated Algorithm Configuration*. 2022. DOI: 10.48550/ARXIV.2202.01651. URL: https://arxiv.org/abs/2202.01651 (pg. 88).

[Sch14] Schauerte, Boris. *Microsoft Academic: conference field ratings*. http://www.conferenceranks.com/visualization/msar2014.html. 2014 (pg. 133).

[Scu10a] Sculley, D. "Web-Scale k-Means Clustering". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: Association for Computing Machinery, 2010, pp. 1177–1178. ISBN: 9781605587998. DOI: 10.1145/1772690.1772862 (pg. 20).

[Scu10b] Sculley, D. "Web-Scale k-Means Clustering". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: Association for Computing Machinery, 2010, pp. 1177–1178. ISBN: 9781605587998. DOI: 10.1145/1772690.1772862. URL: https://doi.org/10.1145/1772690.1772862 (pg. 90).

[SD18] Silva, Jaime A Teixeira da and Dobránszki, Judit. "Multiple versions of the h-index: Cautionary use for formal academic purposes". In: *Scientometrics* 115.2 (2018), pp. 1107–1113 (pg. 115, 117).

[Sek08] Sekercioglu, Cagan H. "Quantifying Coauthor Contributions". In: *Science* 322.5900 (2008), pp. 371–371. ISSN: 0036-8075. DOI: 10.1126/science.322.5900.371a. URL: http://science.sciencemag.org/content/322/5900/371.1 (pg. 128).

[Sha+18] Shah, Nihar B., Tabibian, Behzad, Muandet, Krikamol, Guyon, Isabelle, and Luxburg, Ulrike von. "Design and Analysis of the NIPS 2016 Review Process". In: *JMLR* 19.49 (2018), pp. 1–34. URL: http://jmlr.org/papers/v19/17-511.html (pg. 120).

[She94] Shewchuk, Jonathan Richard. *An introduction to the conjugate gradient method without the agonizing pain*. Tech. rep. Carnegie Mellon University, 1994. URL: https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf (pg. 77).

[Sho57] Shockley, William. "On the statistics of individual variations of productivity in research laboratories". In: *Proceedings of the IRE* 45.3 (1957), pp. 279–290 (pg. 117).

[SI12] Szécsi, László and Illés, Dávid. "Real-Time Metaball Ray Casting with Fragment Lists". In: *Eurographics*. 2012 (pg. 9, 46).

[Sil10] Sill, Joseph. "Improved NBA adjusted+/-using regularization and out-of-sample testing". In: *MIT Sloan Sports Analytics Conference*. 2010 (pg. 140).

[SM16] Srivastava, S. and Michael, N. "Approximate continuous belief distributions for precise autonomous inspection". In: *IEEE SSRR*. Oct. 2016, pp. 74–80. DOI: 10.1109/SSRR.2016.7784280 (pg. 75).

[SM17] Silva, Jaime A Teixeira da and Memon, Aamir Raoof. "CiteScore: A cite for sore eyes, or a valuable, transparent metric?" In: *Scientometrics* 111.1 (2017), pp. 553–556 (pg. 115).

[SM20] Shankar, Kumar Shaurya and Michael, Nathan. "MRFMap: Online Probabilistic 3D Mapping using Forward Ray Sensor Models". In: *Robotics: Science and Systems*. 2020. arXiv: 2006.03512 [cs.RO] (pg. 9, 46).

[SML12] Stoyanov, T., Magnusson, M., and Lilienthal, A. J. "Point set registration through minimization of the L2 distance between 3D-NDT models". In: *ICRA*. May 2012, pp. 5196–5201. DOI: 10.1109/ICRA.2012.6224717 (pg. 76).

[SN01] Schmidt, Jochen and Niemann, Heinrich. "Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization". In: *VMV*. VMV '01. Aka GmbH, 2001, pp. 399–406. ISBN: 3-89838-028-9. URL: http://dl.acm.org/citation.cfm?id=647260.718651 (pg. 77).

[SN60] Selfridge, Oliver G. and Neisser, Ulric. "Pattern Recognition by Machine". In: *Scientific American* 203.2 (1960), pp. 60–69. URL: https://www.jstor.org/stable/24940576 (pg. 2).

[SO94] Schyns, Philippe G. and Oliva, Aude. "From Blobs to Boundary Edges: Evidence for Time- and Spatial-Scale-Dpendent Scene Recognition". In: *Psychological Science* 5.4 (1994), pp. 195–200. URL: http://www.jstor.org/stable/40063101 (pg. 8).

[Spe04] Spearman, Charles. "The proof and measurement of association between two things". In: *The American journal of psychology* 15.1 (1904), pp. 72–101 (pg. 133).

[Sri14] Sridharan, Ramesh. "Gaussian mixture models and the EM algorithm". In: (2014). https://people.csail.mit.edu/rameshvs/content/gmm-em.pdf (pg. 70).

[SSS74] Sutherland, Ivan E., Sproull, Robert F., and Schumacker, Robert A. "A Characterization of Ten Hidden-Surface Algorithms". In: *ACM Comput. Surv.* 6.1 (Mar. 1974), pp. 1–55. ISSN: 0360-0300. DOI: 10.1145/356625.356626 (pg. 3, 13).

[Sta14] Stanfill, Bryan. "Statistical methods for random rotations". PhD thesis. Iowa State University, 2014 (pg. 27).

[Sto+12] Stoyanov, Todor, Magnusson, Martin, Andreasson, Henrik, and Lilienthal, Achim J. "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations". In: *IJRR* 31.12 (2012), pp. 1377–1393. DOI: 10.1177/0278364912460895. URL: https://doi.org/10.1177/0278364912460895 (pg. 76).

[Stu+12] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *IROS*. 2012 (pg. 78).

[Suc+21] Sucar, Edgar, Liu, Shikun, Ortiz, Joseph, and Davison, Andrew. "iMAP: Implicit Mapping and Positioning in Real-Time". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2021 (pg. 152).

[Sut63] Sutherland, Ivan Edward. "Sketchpad, a man-machine graphical communication system". PhD thesis. Massachusetts Institute of Technology, 1963. URL: http://dspace.mit.edu/handle/1721.1/14979 (pg. 3).

[Sut66] Sutherland, Ivan E. "Computer Inputs and Outputs". In: *Scientific American* 215.3 (1966), pp. 86–99. URL: http://www.jstor.org/stable/24931048 (pg. 3).

[Sut70] Sutherland, Ivan E. "Computer Displays". In: *Scientific American* 222.6 (1970), pp. 56–81. URL: http://www.jstor.org/stable/24925827 (pg. 3).

[Tal+16] Tallavajhula, Abhijeet, Choudhury, Sanjiban, Scherer, Sebastian, and Kelly, Alonzo. "List prediction applied to motion planning". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 213–220. DOI: 10.1109/ICRA.2016.7487136 (pg. 87).

[Tan+23] Tancik, Matthew et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*. ACM, 2023. DOI: 10.1145/3588432.3591516. URL: https://doi.org/10.1145%2F3588432.3591516 (pg. 44).

[TD20] Teed, Zachary and Deng, Jia. "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow". English (US). In: *16th European Conference on Computer Vision*. Germany, 2020, pp. 402–419. DOI: 10.1007/978-3-030-58536-5_24 (pg. 7).

[TD21a] Teed, Zachary and Deng, Jia. *DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras*. 2021. arXiv: 2108.10869 [cs.CV] (pg. 7).

[TD21b] Teed, Zachary and Deng, Jia. *Tangent Space Backpropagation for 3D Transformation Groups*. 2021. arXiv: 2103.12032 [cs.CV] (pg. 34).

[Tew+17] Tewari, A., Zollhöfer, M., Kim, H., Garrido, P., Bernard, F., Pérez, P., and Theobalt, C. "MoFA: Model-Based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3735–3744. DOI: 10.1109/ICCV.2017.401 (pg. 9).

[Tho33] Thompson, William R. "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". In: *Biometrika* 25.3/4 (1933), pp. 285–294. ISSN: 00063444. URL: http://www.jstor.org/stable/2332286 (visited on 09/15/2022) (pg. 91).

[TO99] Turk, Greg and O'Brien, James F. *Variational Implicit Surfaces*. Tech. rep. GIT-GVU-99-15. Georgia Institute of Technology, 1999. URL: http://hdl.handle.net/1853/3382 (pg. 148).

[Tol08] Tol, Richard S. J. "A rational, successive g-index applied to economics departments in Ireland". In: *J. Informetrics* 2 (2008), pp. 149–155 (pg. 116).

[Tom+12] Tomic, Teodor, Schmid, Korbinian, Lutz, Philipp, Domel, Andreas, Kassecker, Michael, Mair, Elmar, Grixa, Iris Lynne, Ruess, Felix, Suppa, Michael, and Burschka, Darius. "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue". In: *IEEE Robotics Automation Magazine* 19.3 (2012), pp. 46–56. DOI: 10.1109/MRA.2012.2206473 (pg. 7).

[TOM18] Tabib, W., O'Meadhra, C., and Michael, N. "On-Manifold GMM Registration". In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018), pp. 3805–3812. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2856279 (pg. 9, 14, 67, 75, 77–79, 85).

[Tou+23] Touvron, Hugo et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL] (pg. 108).

[Tri+00] Triggs, Bill, McLauchlan, Philip F., Hartley, Richard I., and Fitzgibbon, Andrew W. "Bundle Adjustment — A Modern Synthesis". In: *Vision Algorithms: Theory and Practice*. Ed. by Triggs, Bill, Zisserman, Andrew, and Szeliski, Richard. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372. ISBN: 978-3-540-44480-0 (pg. 19).

[TS20] Tucker, Richard and Snavely, Noah. "Single-view view synthesis with multiplane images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 551–560 (pg. 8).

[TSG19] Tsai, C, Sankaranarayanan, A, and Gkioulekas, I. "Beyond Volumetric Albedo". In: *CVPR*. June 2019 (pg. 23, 152).

[UV61] Uhr, Leonard and Vossler, Charles. "A Pattern Recognition Program That Generates, Evaluates, and Adjusts Its Own Operators". In: *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*. IRE-AIEE-ACM '61 (Western). Los Angeles, California: Association for Computing Machinery, 1961, pp. 555–569. ISBN: 9781450378727. DOI: 10.1145/1460690.1460751 (pg. 2).

[VHE15] Valenzuela, Marco, Ha, Vu, and Etzioni, Oren. "Identifying meaningful citations". In: *AAAI Workshop: Scholarly Big Data*. 2015 (pg. 116, 128, 129, 133, 134).

[VJK21] Vicini, Delio, Jakob, Wenzel, and Kaplanyan, Anton. "A Non-Exponential Transmittance Model for Volumetric Scene Representations". In: *ACM*

Trans. Graph. 40.4 (July 2021). ISSN: 0730-0301. DOI: 10.1145/3450626.3459815. URL: https://doi.org/10.1145/3450626.3459815 (pg. 49).

[VL99] Vasconcelos, Nuno and Lippman, Andrew. "Learning Mixture Hierarchies". In: *NIPS*. Ed. by Kearns, M. J., Solla, S. A., and Cohn, D. A. MIT Press, 1999, pp. 606–612. URL: http://papers.nips.cc/paper/1543-learning-mixture-hierarchies.pdf (pg. 75).

[Vuc+18] Vucetic, Slobodan, Chanda, Ashis Kumar, Zhang, Shanshan, Bai, Tian, and Maiti, Aniruddha. "Peer Assessment of CS Doctoral Programs Shows Strong Correlation with Faculty Citations". In: *Commun. ACM* 61.9 (Aug. 2018), pp. 70–76. ISSN: 0001-0782 (pg. 116, 117, 119, 131, 132, 134).

[Wal17] Walters, W. H. "Citation-Based Journal Rankings: Key Questions, Metrics, and Data Sources". In: *IEEE Access* 5 (2017), pp. 22036–22053. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2761400 (pg. 117).

[Wan+23] Wang, Angtian, Wang, Peng, Sun, Jian, Kortylewski, Adam, and Yuille, Alan. "VoGE: A Differentiable Volume Renderer using Gaussian Ellipsoids for Analysis-by-Synthesis". In: *The Eleventh International Conference on Learning Representations*. 2023 (pg. 9, 46, 47, 49, 52, 63).

[War69] Warnock, John Edward. "A Hidden Surface Algorithm for Computer Generated Halftone Pictures". PhD thesis. The University of Utah, 1969. URL: https://collections.lib.utah.edu/ark:/87278/s6vj1cn7 (pg. 3).

[Wat70] Watkins, Gary Scott. "A Real Time Visible Surface Algorithm". PhD thesis. The University of Utah, 1970. URL: https://apps.dtic.mil/sti/citations/AD0762004 (pg. 3).

[Wes90] Westover, Lee. "Footprint Evaluation for Volume Rendering". In: *SIGGRAPH Comput. Graph.* 24.4 (Sept. 1990), pp. 367–376. ISSN: 0097-8930. DOI: 10.1145/97880.97919. URL: https://doi.org/10.1145/97880.97919 (pg. 46).

[WF07] Williams, Oliver and Fitzgibbon, Andrew. "Gaussian Process Implicit Surfaces". In: *Gaussian Processes in Practice*. Apr. 2007. URL: https://www.microsoft.com/en-us/research/publication/gaussian-process-implicit-surfaces-2/ (pg. 148).

[WGK20] Westman, E., Gkioulekas, I., and Kaess, M. "Volumetric Albedo Framework for 3D Imaging Sonar". In: *ICRA*. 2020 (pg. 23, 150).

[WMW86] Wyvill, Geoff, McPheeters, Craig, and Wyvill, Brian. "Data structure forsoft objects". In: *The Visual Computer* 2.4 (Aug. 1986), pp. 227–234. ISSN: 1432-2315. DOI: 10.1007/BF01900346 (pg. 9, 46).

[Wro21] Wronski, Bartlomiej. *Procedural Kernel Networks*. 2021. DOI: 10.48550/ARXIV.2112.09318. URL: https://arxiv.org/abs/2112.09318 (pg. 88).

[WT90] Wyvill, Geoff and Trotman, Andrew. "Ray-Tracing Soft Objects". In: *CG International*. Ed. by Chua, Tat-Seng and Kunii, Tosiyasu L. Tokyo: Springer Japan, 1990, pp. 469–476. ISBN: 978-4-431-68123-6 (pg. 9, 46).

[Wu+19] Wu, Yuxin, Kirillov, Alexander, Massa, Francisco, Lo, Wan-Yen, and Girshick, Ross. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019 (pg. 24).

[XHL10] Xu, Lin, Hoos, Holger H., and Leyton-Brown, Kevin. "Hydra: Automatically Configuring Algorithms for Portfolio-Based Selection". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI'10. Atlanta, Georgia: AAAI Press, 2010, pp. 210–216 (pg. 88).

[Xu+23] Xu, Haofei, Zhang, Jing, Cai, Jianfei, Rezatofighi, Hamid, Yu, Fisher, Tao, Dacheng, and Geiger, Andreas. "Unifying Flow, Stereo and Depth Estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023) (pg. 50, 53–55).

[XXM16] Xu, D., Xia, Y., and Mandic, D. P. "Optimization in Quaternion Dynamic Systems: Gradient, Hessian, and Learning Algorithms". In: *IEEE Transactions on Neural Networks and Learning Systems* 27.2 (Feb. 2016), pp. 249–261. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2015.2440473 (pg. 77).

[Yan+21a] Yang, Gengshan, Sun, Deqing, Jampani, Varun, Vlasic, Daniel, Cole, Forrester, Chang, Huiwen, Ramanan, Deva, Freeman, William T, and Liu, Ce. "LASR: Learning Articulated Shape Reconstruction from a Monocular Video". In: *CVPR*. June 2021 (pg. 52, 152).

[Yan+21b] Yang, Gengshan, Sun, Deqing, Jampani, Varun, Vlasic, Daniel, Cole, Forrester, Liu, Ce, and Ramanan, Deva. "ViSER: Video-Specific Surface Embeddings for Articulated 3D Shape Reconstruction". In: *NeurIPS*. Dec. 2021 (pg. 52, 152).

[Yan+22] Yang, Gengshan, Vo, Minh, Neverova, Natalia, Ramanan, Deva, Vedaldi, Andrea, and Joo, Hanbyul. "BANMo: Building Animatable 3D Neural Models from Many Casual Videos". In: *CVPR*. 2022 (pg. 152).

[Yan+23] Yang, Gengshan, Yang, Shuo, Zhang, John Z., Manchester, Zachary, and Ramanan, Deva. "Physically Plausible Reconstruction from Monocular Videos". In: *ICCV*. 2023 (pg. 152).

[YDB21] Ye, Furong, Doerr, Carola, and Bäck, Thomas. "Leveraging Benchmarking Data for Informed One-Shot Dynamic Algorithm Selection". In: *Proceedings of the Genetic and Evolutionary Computation Confer-*

ence Companion. GECCO '21. Lille, France: Association for Computing Machinery, 2021, pp. 245–246. ISBN: 9781450383516. DOI: 10.1145/3449726.3459578 (pg. 88).

[Yif+19] Yifan, Wang, Serena, Felice, Wu, Shihao, Öztireli, Cengiz, and Sorkine-Hornung, Olga. "Differentiable surface splatting for point-based geometry processing". In: *ACM Transactions on Graphics* 38.6 (Nov. 2019), pp. 1–14. ISSN: 1557-7368. DOI: 10.1145/3355089.3356513 (pg. 7, 8, 16, 17, 20, 27, 32, 44, 46).

[YL07] Yan, Su and Lee, Dongwon. "Toward Alternative Measures for Ranking Venues: A Case of Database Research Community". In: *7th ACM/IEEE-CS Joint Conference on Digital Libraries*. JCDL '07. Vancouver, BC, Canada: ACM, 2007, pp. 235–244. ISBN: 978-1-59593-644-8 (pg. 117, 131).

[YLJ13] Yang, Jiaolong, Li, Hongdong, and Jia, Yunde. "Go-ICP: Solving 3D Registration Efficiently and Globally Optimally". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 1457–1464. DOI: 10.1109/ICCV.2013.184 (pg. 18).

[Yon14] Yong, Alexander. "Critique of Hirsch's citation index: A combinatorial Fermi problem". In: *Notices of the AMS* 61.9 (2014), pp. 1040–1050 (pg. 133).

[Yoo+22] Yoon, Jaehong, Madaan, Divyam, Yang, Eunho, and Hwang, Sung Ju. "Online Coreset Selection for Rehearsal-based Continual Learning". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=f9D-5WNG4Nv (pg. 88).

[YS19] Yang, Shichao and Scherer, Sebastian. "CubeSLAM: Monocular 3-D Object SLAM". In: *IEEE Transactions on Robotics* 35.4 (2019), pp. 925–938. DOI: 10.1109/TRO.2019.2909168 (pg. 7).

[Zar35] Zariski, Oscar. *Algebraic Surfaces*. McGraw Hill, 1935. DOI: 10.1007/978-3-642-61991-5 (pg. 5).

[Zha+17] Zhang, Juanjuan, Fiers, Pieter, Witte, Kirby A., Jackson, Rachel W., Poggensee, Katherine L., Atkeson, Christopher G., and Collins, Steven H. "Human-in-the-loop optimization of exoskeleton assistance during walking". In: *Science* 356.6344 (2017), pp. 1280–1284. DOI: 10.1126/science.aal5054. eprint: https://www.science.org/doi/pdf/10.1126/science.aal5054. URL: https://www.science.org/doi/abs/10.1126/science.aal5054 (pg. 90, 109).

[Zha+20] Zhang, Cheng, Miller, Bailey, Yan, Kai, Gkioulekas, Ioannis, and Zhao, Shuang. "Path-Space Differentiable Rendering". In: *ACM Trans. Graph.* 39.4 (2020), 143:1–143:19. DOI: 10.1145/3386569.3392383 (pg. 6, 8).

[Zha+23] Zhang, Xiaoshuai, Kundu, Abhijit, Funkhouser, Thomas, Guibas, Leonidas, Su, Hao, and Genova, Kyle. "Nerflets: Local Radiance Fields for Efficient Structure-Aware 3D Scene Representation from 2D Supervision". In: *CVPR* (2023) (pg. 46).

[Zha04] Zhang, Tong. "Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms". In: *ICML*. Banff, Alberta, Canada: ACM, 2004, pp. 116–. ISBN: 1-58113-838-5 (pg. 122).

[Zho+21] Zhong, Ellen D., Lerer, Adam, Davis, Joseph H., and Berger, Bonnie. "CryoDRGN2: Ab Initio Neural Reconstruction of 3D Protein Structures From Real Cryo-EM Images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 4066–4075 (pg. 23).

[ZJ16] Zhou, Qingnan and Jacobson, Alec. "Thingi10K: A Dataset of 10, 000 3D-Printing Models". In: *CoRR* abs/1605.04797 (2016). arXiv: 1605.04797. URL: http://arxiv.org/abs/1605.04797 (pg. 14, 27, 55).

[ZPK18] Zhou, Qian-Yi, Park, Jaesik, and Koltun, Vladlen. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018) (pg. 15, 18, 19, 21).

[ZW18] Zhang, Fang and Wu, Shengli. "Ranking Scientific Papers and Venues in Heterogeneous Academic Networks by Mutual Reinforcement". In: *JCDL*. Fort Worth, Texas, USA: ACM, 2018, pp. 127–130. ISBN: 978-1-4503-5178-2 (pg. 117, 131).

[Zwi+01] Zwicker, Matthias, Pfister, Hanspeter, Baar, Jeroen van, and Gross, Markus. "Surface Splatting". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. 2001, pp. 371–378. DOI: 10.1145/383259.383300 (pg. 8).