Ph.D. Thesis
Doctor of Philosophy

DTU Compute
Department of Applied Mathematics and Computer Science

Probabilistic 3D Reconstruction

Frederik Rahbæk Warburg

Kongens Lyngby 2023

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

Deep learning has emerged as a powerful paradigm to create accurate 3D maps for autonomous agents. This 3D awareness enable robots to navigate and interact with the world. However, the deep learning methods to create such maps often come without any notion of uncertainties, which can have catastrophic consequences if wrong predictions are propagated through the system.

This thesis explores the de-facto pipeline for 3D reconstruction and how uncertainties can be incorporated into each step to ensure the safe deployment of autonomous agents.

More specifically, we present a new mental model for understanding uncertainties in deep learning, namely *learned* versus *deduced* uncertainties, that originates in pragmatic considerations and offers practical guidelines on how to model uncertainties. The thesis chronologically makes improvements to the four steps of the reconstruction pipeline: *1. Retrieval.* We present a Bayesian training procedure to deduce uncertainties for stochastic representations that reduces the risk of silent errors in image retrieval. *2. Structure from Motion.* We propose a detector-agnostic method to estimate the uncertainties of deep keypoint detectors and show that the deduced uncertainties improve camera localization accuracy. *3. Multiview Stereo.* We present a factorization of dynamic 3D maps that is memory efficient and enable fast training and rendering. Further, we present a probabilistic model to distill a learned 3D prior of local shapes into the reconstruction process. *4. 3D reasoning.* Last, we present a novel framework to interact with and manipulate 3D maps in a semantically consistent manner.

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a Ph.D. degree in Computer Science.

Kongens Lyngby, November 28, 2023

[illegible]

Frederik Rahbæk Warburg

Acknowledgements

In my Ph.D. I have been fortunate to travel the world and meet so many wonderful, inspiring, and smart people, who have taught me so many things and made the last years stimulating and fun.

First of all, I would like to thank my supervisors. Søren Hauberg for your uncanny intuition on abstract concepts and spaces, for giving me the freedom to pursue my own ideas, for supporting my many travels, and for always being available with feedback. Javier Civera for getting me started with research, inspiring me to pursue a Ph.D., and for our many conversations on computer vision. Serge Belongie for inspiring conversations, encouraging me to pursue crazy ideas, and for always asking *why not just...?*. Angjoo Kanazawa for your passion, eagerness to understand and adapt new technologies, and your consistent curiosity.

I want to thank all my colleagues and collaborators during my Ph.D., who inspired me and made the journey enjoyable. Especially, Marco Miani for your patience in explaining me math, and Ethan Weber for our fun and productive collaborations. I also want to thank a number of colleagues I was fortunate to learn from and collaborate with. In alphabetic order: Aasa Feragen, Aleksander Hołyński, Alireza Kashani, Andrea Vallone, Benjamin Recht, Daniel Hernandez-Juarez, Giacomo Meanti, Grethe Hyldig, Hans Hansen, Javier Tirado-Garin, Jes Frellsen, Juan Tarrio, Juston Moore, Kilian Zepf, Kristjan Eldjarn, Kristoffer H. Madsen, Manuel López-Antequera, Martin Jørgensen, Matt Tancik, Menglin Jia, Michael Ramamonjisoa, Nicki Skafte Detlefsen, Pablo F. Alcantarilla, Pablo Moreno-Muñoz, Pau Gargallo, Peter Ebert Christensen, Pola Schwöbel, Sagie Benaim, Sara Fridovich-Keil, Silas Brack, Simon Moe Søresen, Thoranna Bender, Ujwal Bonde, Yubin Kuang.

Most importantly, I would like to thank friends and family, who kept the conversation scholarly only when it had to. Especially, I want to thank my girlfriend, Mati, for our amazing adventures and for keeping me alive and motivated throughout this journey.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Opening our eyes and perceiving the world around us is a fundamental ability that most of us often take for granted. At a glance, we capture a myriad of visual information, effortlessly discerning objects, people, and their movements. For instance, consider the image pair in Figure 1.1, we see that the church stands behind a building, individuals traverse in one direction, and the camera moves in the opposite direction. Such observations highlight the remarkable precision and spatial awareness of our perception. Its evolutionary importance stems from its accuracy at long distances which has prevented humans from getting too close to dangerous animals and navigating around obstacles [Pal99]. When we perceive the world, we store a spatial three-dimensional (3D) representation of it. Evidence of this internal 3D representation has been found in psychological studies, e.g. consider the objects displayed in Figure 1.2. Our perception system sees a cylinder, a circle, and a cube, layering the objects and completing their shapes, thus implicitly constructing a 3D representation of the scene from only the visible parts of the objects [Pal99]. Storing a 3D canonical representation is statistical and computationally much more efficient compared to recomputing and reprocessing all the information when we move around [Pal99]. Therefore, it is reasonable to assume that if we want robots to navigate the world, we need 3D-aware models and representations.

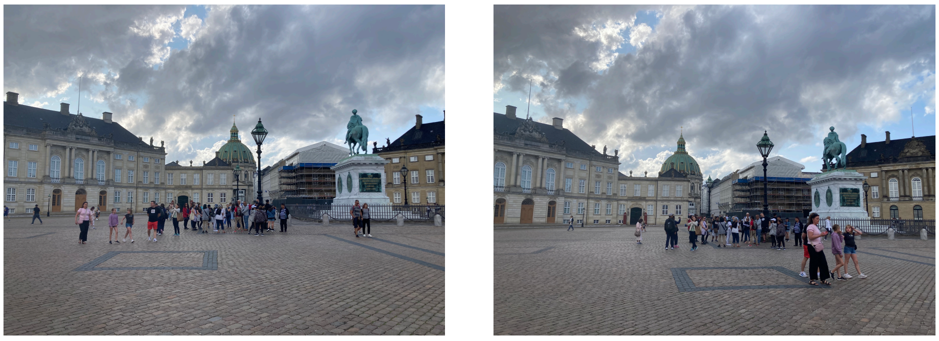


Figure 1.1: We can effortlessly perceive the 3D geometry of the scene and the movement of the camera and the pedestrians.

Figure 1.2: Visual completion behind partly occluding objects. We perceive the left figure as a square, a rectangle, and a circle, although, we only see the visible regions displayed in the right figure.

In recent years, deep learning has emerged as a powerful paradigm, demonstrating remarkable accomplishments across various domains [Mil+20; Rom+21; FC20]. Also, within the field of computer vision, deep learning has exhibited significant potential for creating accurate 3D maps. Leveraging deep learning for 3D representations has enabled many applications, including robotic navigation and augmented reality. These methods are currently deployed without or with a limited notion of uncertainty. However, deep learning methods are not perfect. Wrong predictions will happen. And if these errors are propagated through the reconstruction pipeline, they can have devastating consequences. One example was when a perception system in an autonomous vehicle predicted the side of a white truck to be a bright sky. This led to the first fatal accident in an autonomous vehicle [Dep17]. Had the system been able to assign higher uncertainty to its erroneous prediction, the incident might have been avoided.

This thesis aims to push the boundaries of modeling uncertainties in deep learning and improve the state-of-the-art in 3D reconstruction. By addressing these challenges, this thesis seeks to enhance the reliability, efficiency, and accuracy of deep learning-based spatial perception systems. Overall, this research represents a step towards advancing the field of computer vision, enhancing the capabilities of spatial perception systems, and paving the way for safer and more reliable applications such as autonomous vehicles, robotics, and mixed reality.

1.2 Approach

Humans and machines alike only observe a 2D projection of the 3D world. Reconstructing the 3D scene from a single 2D image is an ill-posed problem. However, the human perception system gives evidence that it is feasible to solve efficiently and reliably. We know that the human perception system uses many cues to inform us about the underlying 3D scenes, ranging from multiview stereo, shape from shading, and shape and scale from object semantics [Pal99].

The computer vision community has studied these cues for decades and developed sophisticated methods to infer the 3D geometry of scenes and objects. The most common and well-studied practical pipeline for 3D reconstruction is illustrated in Figure 1.3 and consists of first retrieving the images of an object or scene to reconstruct, then running Structure from Motion (SfM) to estimate the camera poses, and

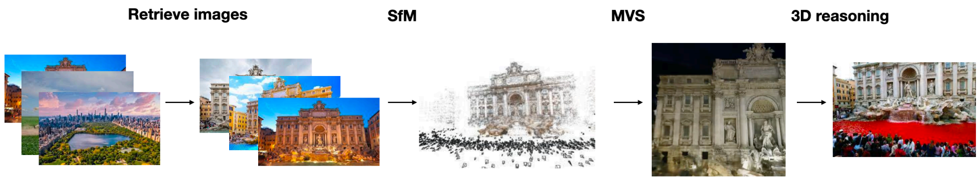


Figure 1.3: **From 2D images to 3D.** Given a collection of unstructured images, the first step is to retrieve the images to use for the 3D reconstruction. Then Structure from Motion (SfM) is run to find get posed images and the sparse 3D geometry. We can obtain a dense 3D reconstruction with Multiview Stereo (MVS). Once we have a dense 3D reconstruction, we can edit the scene, e.g. how would the Trevi fountain look if the water was red?

finally using Multi-View Stereo (MVS) to obtain a dense 3D reconstruction on which 3D reasoning can be performed.

This thesis contributes to each step of the pipeline, focusing on advancing the state-of-the-art in dense 3D reconstruction, while modeling uncertainties at each stage of the pipeline. This research seeks to enhance our understanding of uncertainty in 3D reconstruction and contribute to the development of more accurate and trustworthy spatial perception systems.

1.3 Contributions

To summarise, the contributions of this thesis are as follows

- We develop a Bayesian algorithm to model uncertainties in representation learning and image retrieval.
- We demonstrate how to quantify uncertainties for visual localization.
- We present an efficient method to represent dynamic 3D reconstructions.
- We show how generative priors can be incorporated into the reconstruction pipeline.
- Finally, we present a method for disentangling and manipulating 3D scenes.

1.4 Co-Authored Papers

Some extracts from this thesis appear in the following co-authored publications and preprints. Chapter 2 is extended from:

- Laplacian Autoencoders for Learning Stochastic Representations [Mia+22] (NeurIPS 2022)

Chapter 3 is adapted from:

- Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition [War+20] (CVPR 2020)
- Bayesian Metric Learning for Uncertainty Quantification in Image Retrieval [War+23a] (NeurIPS 2023)

Chapter 4 builds on:

- DAC: Detector-Agnostic Spatial Covariances for Deep Local Features [TWC23] (preprint 2023)

Chapter 5 is adapted from

- Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs [War+23b] (ICCV 2023)
- K-Planes: Explicit Radiance Fields in Space, Time, and Appearance [Fri+23] (CVPR 2023)

Lastly, Chapter 6 summarizes

- Volumetric Disentanglement for 3D Scene Manipulation [Ben+22] (preprint 2022)

The following papers were written during the Ph.D., but are not included in the thesis

- Learning to Taste: A Multimodal Wine Dataset [Ben+23] (NeurIPS 2023)
- Searching for Structure in Unfalsifiable Claims [Chr+22] (HCOMP 2022)
- Bayesian Triplet Loss: Uncertainty Quantification in Image Retrieval [War+21] (ICCV 2021)
- Danish Airs and Grounds: A dataset for aerial-to-street-level place recognition and localization [Val+22] (IROS 2022)
- Probabilistic Spatial Transformer Networks [Sch+22] (UAI 2022)
- Laplacian Segmentation Networks: Improved Epistemic Uncertainty from Spatial Aleatoric Uncertainty [Zep+23] (preprint 2023)
- Self-Supervised Depth Completion for Active Stereo [War+22] (ICRA 2022)
- SparseFormer: Attention-based Depth Completion Network [WRL22] (CV4ARVR 2022)

1.5 Thesis Structure

Chapter 2 presents the de-facto division of uncertainties, e.g. epistemic and aleatoric, and proposes an alternative framework based on learned versus deduced uncertainties. The rest of the thesis follows the four steps illustrated in Figure 1.3. Chapter 3 shows that deduced uncertainties are useful for out-of-distribution examples in representation learning and image retrieval. Chapter 4 that deduced uncertainties provide reliable uncertainties for visual localization. Chapter 5 presents work on incorporating generative priors into multi-view stereo pipelines and modeling dynamic 3D reconstructions. Lastly, Chapter 6 proposes a framework to interact with and manipulate implicit 3D reconstructions.

CHAPTER 2 Uncertainties in Deep Learning

The uncertainty of a model's predictions arises from either aleatoric (data) uncertainty or epistemic (model) uncertainty [DD09]. This is currently the de-facto mental model for understanding uncertainties in deep learning [KG17]. The epistemic uncertainty arises because the model's parameters are estimated from a finite amount of data, and there is inherent uncertainty about the optimal values for these parameters. In contrast, aleatoric uncertainty stems from the inherent randomness in the data generation process. It represents the uncertainty that remains even if we have a perfect model with precise knowledge of its parameters.

Although this mental model is nice-to-have, in practice, the disentanglement of the types of uncertainties is challenging for neural networks [Wim+23; VM22]. Moreover, we can often only evaluate the predictive uncertainty (e.g. the union of the two uncertainties). Hence, the disentanglement has limited practical value. This chapter challenges the de-facto division of uncertainties and proposes an alternative division into *learned* versus *deduced* uncertainties. This division offers more practical guidelines on how to model uncertainties in neural networks. We further present a novel online training process of the Laplace Approximation (LA) and an efficient Hessian approximation that enables us to estimate reliable uncertainties in large neural networks with high dimensional outputs.

This chapter contains parts from Laplacian Autoencoders for Learning Stochastic Representations [Mia+22], Bayesian Metric Learning for Uncertainty Quantification in Image Retrieval [War+23a], and pytorch-laplace [WM23]. Not all experiments, results, and related works are presented to keep the text shorter.

2.1 Epistemic and Aleatoric Uncertainties

Epistemic Uncertainty. In Bayesian deep learning, epistemic uncertainty is often addressed by placing a prior distribution over the model's parameters and updating this distribution based on the observed data to obtain the posterior distribution. The posterior distribution captures our updated knowledge about the parameters given the data. Mathematically, let's denote the model's parameters as θ and the data as D. The epistemic uncertainty can be represented by the posterior distribution $p(\theta|D)$ over the model parameters θ conditioned on the observed data D.

Aleatoric Uncertainty. The aleatoric uncertainty can be captured by introducing a probability distribution over the target values y for each input data point x. This means that instead of predicting a single value, the model provides a probability distribution over possible target values. The aleatoric uncertainty can be represented by the conditional distribution $p(y|x)$, which captures the variability in the target variable y given the input data x.

Predictive Uncertainty. In practice, a deep learning model may exhibit both epistemic and aleatoric uncertainties simultaneously. This is often referred to as predictive uncertainty, and it can be modeled using Bayesian neural networks that incorporate both epistemic and aleatoric uncertainty components. The predictive distribution can be expressed as follows:

$$p(y|x, D) = \int p(y|x, \theta)p(\theta, D)d\theta \tag{2.1}$$

where $p(y|x, \theta)$ represents the aleatoric uncertainty in the predictions, and $p(\theta|D)$ represents the epistemic uncertainty in the model's parameters. For deep networks, it is hard to disentangle the epistemic and aleatoric parts [Wim+23; VM22], and we can only evaluate the predictive uncertainty. Therefore, in practice, the division primarily serves as a mental model for uncertainties with limited practical value.

2.2 Learned and Deduced Uncertainties

Here, we advocate for an alternative division of uncertainty quantification into *learned* versus *deduced* uncertainties. This division bears parallels to the aleatoric/epistemic one, however, gives rise to a different mental model. It arises from more pragmatic and practical considerations, namely *what do we need the uncertainties for?* For example, a common approach to learning uncertainties is by parameterizing the predictive distribution $p_\theta(y|x) \sim \mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$, where a neural network predicts both the mean and variance estimates. The variance network σ^2_θ usually works well for in-distribution data, where the network operates within the range of the training data. However, it faces challenges when dealing with out-of-distribution data, as the neural network must extrapolate beyond its training data [HG16; VM22]. Neural networks are generally poor at extrapolation, leading to difficulties in assigning high uncertainties to out-of-distribution data accurately.

An alternative to using a neural network for predicting uncertainties is to deduce uncertainties. One such approach is deep ensembles [LPB17], where multiple networks are trained with different random seed initialization. Predictive uncertainties are then obtained by forwarding the data through each model and computing the mean and variance of their predictions. Deep ensembles provide a more reliable approach to quantifying uncertainties for out-of-distribution (OOD) data because they do not rely on the network's ability to extrapolate. The disadvantage of methods that deduce uncertainties, such as deep ensembles, is that they rely on sampling, which adds a computational overhead.

In short, this division of uncertainties tells us that learned uncertainties are useful for in-distribution (ID) data and deduced uncertainties are useful for out-of-distribution (OOD) data. The advantage of this mental model is that it does not give any false promises on disentangling the origin of the predictive uncertainty.

2.3 Laplace Approximation (Posthoc)

The Laplace approximation [Mac92] is a practical method to deduce the uncertainty of a neural network. It assumes that the weights of a neural network follow a Gaussian distribution (Gaussian weight-posterior), which is estimated using the curvature of the loss landscape.

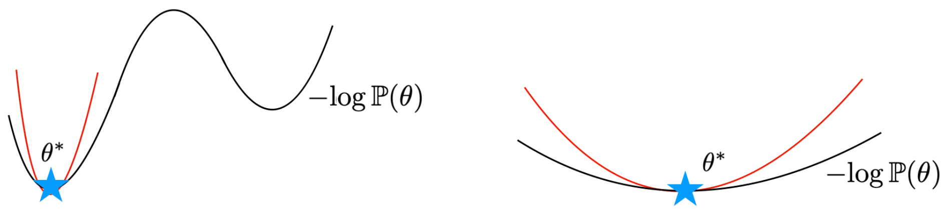


Figure 2.1: **Intuition of Laplace Approximation**. Illustrates two loss landscapes; one with a steep valley (left) and one more flat (right), their local minima (star), and a second-order Taylor expansion around the minimas.

Intuition. Figure 2.1 shows the loss (or negative log-likelihood) for two one-parameter neural networks. After training, the parameters converge to a local minimum of the loss, illustrated by the blue star θ^*. If a parameter is in a steep valley, changing it a little bit will increase the loss a lot. This means that we are fairly certain about the specific value of the parameter. On the other hand, if a parameter is in a flat valley, changing it a little bit will not increase the loss a lot. Thus, we are uncertain about the specific value of the parameter.

The steepness of the valley is determined by the second derivative of the loss, also called the Hessian. Thus, the inverse of the Hessian determines the uncertainty of the parameters. In the following, we will derive the Laplace approximation and show that it corresponds to assuming a Gaussian distribution over the parameters (Gaussian weight-posterior).

Derivations. The Laplace approximation (LA) can be applied for every loss function $\mathcal{L}$ that can be interpreted as an unnormalized log-posterior by performing a second-order Taylor expansion around a chosen weight vector θ^*:

$$\log p(\theta \mid \mathcal{D}) = \mathcal{L}^* + (\theta - \theta^*)^\top \nabla \mathcal{L}^* + \frac{1}{2} (\theta - \theta^*)^\top \nabla^2 \mathcal{L}^* (\theta - \theta^*) + \mathcal{O}\left(\|\theta - \theta^*\|^3\right) \tag{2.2}$$

Imposing the unnormalized log-posterior to be a second-order polynomial is equivalent to assuming the posterior to be Gaussian.

If θ^* is the maximum a posteriori (MAP) estimate the gradient is zero, such that the first-order term in the Taylor expansion vanishes, which yields a weight posterior as:

$$p(\theta \mid \mathcal{D}) = \mathcal{N}\left(\theta \mid \theta^*, \left(\nabla_\theta^2 \mathcal{L}(\theta^*; \mathcal{D}) + \sigma_{\text{prior}}^{-2} \mathbb{I}\right)^{-1}\right) \tag{2.3}$$

where $\mathcal{N}$ denotes the Gaussian distribution. The advantage of post-hoc LA is that the training procedure does not change, and already trained neural networks can be made Bayesian. In practice, however, we empirically observe post-hoc LA to be unstable [War+23a]. The instability stems from curvature differences between the local minima through stochastic optimization.

2.4 Laplace Approximation (Online)

We propose an online LA [Mia+22] that improves on this instability by marginalizing the LA during training with Monte Carlo EM. This helps the training recover a solution θ^* where the Hessian reflects the loss landscape. Specifically, at each step t during training, we keep in memory a Gaussian distribution on the parameters $q^t(\theta) = \mathcal{N}(\theta | \theta_t, H_{\theta_t}^{-1})$. The parameters are updated through an expected gradient step

$$\theta_{t+1} = \theta_t + \lambda \mathbb{E}_{\theta \sim q^t} [\nabla_\theta \mathcal{L}(\theta; \mathcal{D})] \tag{2.4}$$

and a discounted Laplace update

$$H_{\theta_{t+1}} = (1 - \alpha) H_{\theta_t} + \nabla_\theta^2 \mathcal{L}(\theta; \mathcal{D}), \tag{2.5}$$

where α describes an exponential moving average, similar to momentum-like training. The initialization follows the isotropic prior $q^0(\theta) = \mathcal{N}(\theta | 0, \sigma_{\text{prior}}^2 \mathbb{I})$, mimicking how neural networks are commonly initialized.

Figure 2.2 compares a varational autoencoder [KW13] (VAE) with a post-hoc and online Laplacian autoencoder [Mia+22] (LAE). The LAE is the Laplace approximated posterior of an AE. We find that (1) the VAE does not exhibit good uncertainties in the latent space. This is because the uncertainties are learned, and thus, do not extrapolate far away from the training data distribution. The LAEs yield reliable uncertainties in the latent space, even far away from the training data. We find (2) that the post-hoc LAE does not yield good uncertainties in the output space, whereas

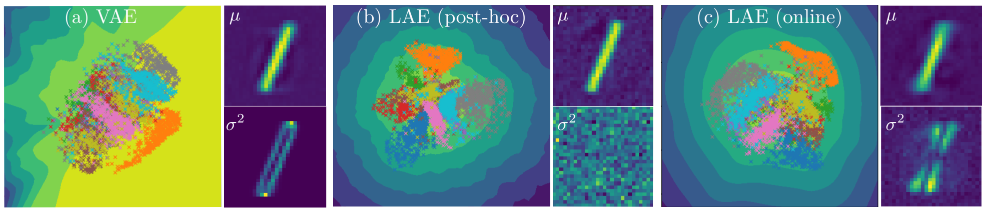


Figure 2.2: 2D latent representation of MNIST overlaid a heatmap that describes the decoder uncertainty (yellow/blue indicates a low/high variance of the reconstructions). To the right of the latent spaces, we show the mean and variance of a reconstructed image (yellow indicates high values). (a) The VAE learns to estimate high variance for low-level image features such as edges but fails at extrapolating uncertainties away from training data. (b) Applying post-hoc Laplace to the AE setup shows much better extrapolating capabilities, but fails in estimating calibrated uncertainties in output space. (c) Our online, sampling-based optimization of a Laplacian autoencoder (LAE) gives well-behaved uncertainties in both latent and output space.

the online LAE is better calibrated, providing reasonable uncertainties in the output space, depicted by the high variance at the top and bottom of the "1" character capturing the variability in which people write.

2.5 Scaling the Hessian to Large Images

The largest obstacle to applying LA in practice stems from the Hessian matrix. This matrix has a quadratic memory complexity in the number of network parameters, which very quickly exceeds the capabilities of available hardware. To counter this issue, several approximations have been proposed [RBB18; BRB17; MG15] that improve the scaling w.r.t. to the number of parameters. The currently most efficient Hessian implementations [DKH20; Dax+21] builds on the generalized Gauss-Newton (GGN) approximation of the Hessian

$$\nabla^2_{\boldsymbol{\theta}^{(l)}} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x})) \approx J_{\boldsymbol{\theta}^{(l)}} f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)^\top \cdot \nabla^2_{\boldsymbol{y}} \mathcal{L}\left(\boldsymbol{y}\right) \cdot J_{\boldsymbol{\theta}^{(l)}} f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right), \tag{2.6}$$

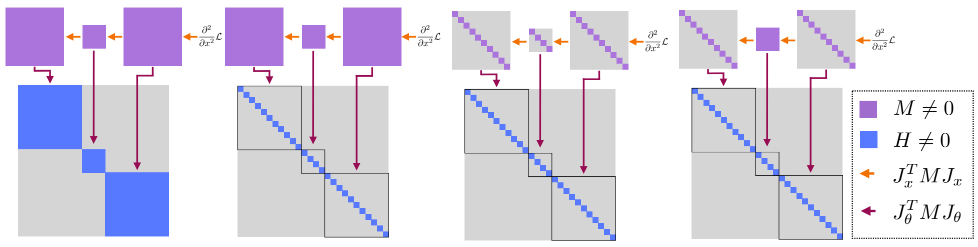
for a single layer l, which neglects second order derivatives of f w.r.t. the parameters. Besides, the computational benefits of this approximation, previous works on LA [Dax+21] rely on GGN to ensure that the Hessian is always semi-negative definite.

Albeit relying on first-order derivates, the layer-block-diagonal GGN, which assumes that layers are independent of each other, scales quadratically with the *output* dimension of the considered neural network f. This lack of scaling is particularly detrimental for convolutional layers as these have low parameter counts, but potentially very high output dimensions. Expanding $J_{\boldsymbol{\theta}^{(l)}} f_{\boldsymbol{\theta}}(\boldsymbol{x})$ with the chain rule, one realizes that the Jacobian can be computed as a function of the Jacobian of the next layer. Figure 2.3 illustrate that an intermediate quantity M, which is initialised as

$\nabla^2_{\boldsymbol{y}}\mathcal{L}(\boldsymbol{y})$, can be efficiently backpropagated through multiplication with the Jacobian w.r.t. input of each layer. This process leads to a **block diagonal** approximation of the Hessian as illustrated in Figure 2.3(a). However, diagonal blocks are generally too large to store and invert. To combat this, each block can be further approximated by its **exact diagonal** [Dax+21] as depicted in Figure 2.3(b). This scales linearly w.r.t. parameters, but still scales quadratically w.r.t. the output resolution (Table 2.1).

APPROXIMATIONS	MEMORY	TIME
Block diag.	$\mathcal{O}(R_m^2 + W_s^2)$	$\mathcal{O}(R_s^2 + W_s^2)$
KFAC	$\mathcal{O}(R_s^2 + W_s)$	$\mathcal{O}(R_s^2 + W_s)$
Exact diag.	$\mathcal{O}(R_m^2 + W_s)$	$\mathcal{O}(R_s^2 + W_s)$
Approx. diag. (ours)	$\mathcal{O}(R_m + W_s)$	$\mathcal{O}(R_s + W_s)$
Mixed diag. (ours)	$\mathcal{O}(R_m + W_s)$	$\mathcal{O}(R_s + W_s)$

Table 2.1: **Memory & time complexity of Hessian approximations**. For an L-layer network, let $R_m{=}\max_{l=0\ldots L}|x^{(l)}|$, $R_s{=}\sum_l |x^{(l)}|$, $R_s^2{=}\sum_l |x^{(l)}|^2$, and $W_s{=}\sum_l |\boldsymbol{\theta}^{(l)}|$. Only our approximation scales linearly with both the output resolution and parameters.



$\frac{\partial^2}{\partial x^2}\mathcal{L}$
$\frac{\partial^2}{\partial x^2}\mathcal{L}$
$\frac{\partial^2}{\partial x^2}\mathcal{L}$
$\frac{\partial^2}{\partial x^2}\mathcal{L}$
$M \neq 0$
$H \neq 0$
$J_x^T M J_x$
$J_\theta^T M J_\theta$

(a) Block diagonal (b) Exact diagonal (c) Approx. diagonal (d) Mixed diagonal

Figure 2.3: **Comparison of Hessian approximation methods.** Common approximations (a–b) scale quadratically with the output resolution. Our proposed approximate and mixed diagonal Hessians (c–d) scale linearly with the resolution. This is essential for scaling the LAE to large images.

To scale our Hessian to high-dimensional data, we propose to approximate the diagonal of the Hessian rather than relying on exact computations. We achieve this by only backpropagating a diagonal form of M as illustrated in Figure 2.3(c). This assumes that features from the same layer are uncorrelated and consequently have linear complexity in both time and memory with respect to the output dimension (Table 2.1). This makes it viable for models with large output dimensions.

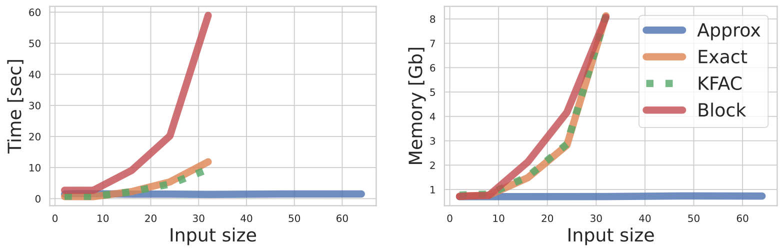


Figure 2.4: **Memory & time usage of Hessian approximations.** The exact and KFAC scales poorly with the image resolution. In contrast, our proposed approximate diagonal Hessian scales linearly.

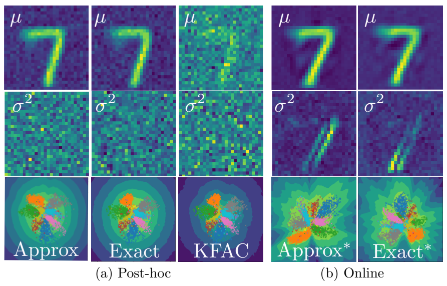


Figure 2.5: Mean and variance of 100 sampled NN.

Hessian	$-\log p(x) \downarrow$	MSE $\downarrow$
KFAC	9683.9 ± 2455.0	121.6 ± 24.5
Exact	283.3 ± 88.6	27.1 ± 0.9
Approx	232.0 ± 65.5	26.6 ± 0.6
Exact*	25.8 ± 0.2	25.7 ± 0.2
Approx*	25.9 ± 0.4	25.8 ± 0.4

Table 2.2: Online training (indicated by *) outperforms post-hoc LA. The approximate diagonal has similar performance to the exact diagonal for both post-hoc and online LA.

We can further tailor this approximation to the autoencoder setting by leveraging the bottleneck architecture. We note that the quadratic scaling of the exact diagonal Hessian is less of an issue in the layers near the bottleneck than in the layers closer to the output space. We can therefore dynamically switch between our approximate diagonal and the exact one, depending on the feature dimension. This lessens the approximation error while remaining tractable in practice.

2.6 Experiments

For practical applications, training time and memory usage of the Hessian approximation must be kept low. We here show that the proposed approximate diagonal Hessian is sufficient and even outperforms other approximations when combined with our online training.

Figure 2.4 show the time and memory requirement for different approximation methods as a function of input size for a 5-layer convolutional network that preserves channel and input dimension. As baselines we use efficient implementations of the exact and KFAC approximation [Dax+21; DKH20]. The exact diagonal approximation run out of memory for an $\sim 36 \times 36 \times 3$ image on a 11 Gb NVIDIA GeForce GTX 1080 Ti. In contrast, our approximate diagonal Hessian scales linearly with the resolution, which is especially beneficial for convolutional layers.

Table 2.2 shows that the exact or approximate Hessian diagonal has similar performance for both post-hoc and online training. Using post-hoc LA results in good mean reconstructions (low MSE), but each sampled NN does not give good reconstructions (low $\log p(x)$). Using our online training procedure results in a much higher log-likelihood. This indicates that every sampled NN predicts good reconstructions.

Figure 2.5 shows the latent representation, mean, and variance of the reconstructions with the KFAC, exact and approximate diagonal for both post-hoc and online setup. Note that the online training makes the uncertainties better fitted, both in latent and data space. These well-fitted uncertainties have several practical downstream applications, such as out-of-distribution detection and semi-supervised learning [Mia+22].

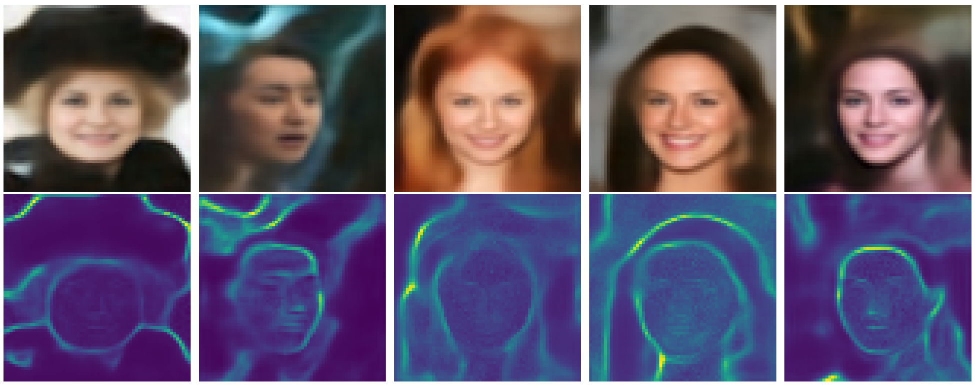


Figure 2.6: **Sample reconstructions on CelebA.** The top row shows the mean reconstruction and the bottom row shows the variance of the reconstructed images.

Figure 2.6 shows the mean and variance of five reconstructed images. The online LAE produces well-calibrated uncertainties in the output space and scales to large images.

2.7 Summary

This chapter challenges the de-facto mental model of uncertainties in the deep learning community, proposes an alternative division of uncertainties into learned versus deduced uncertainties, and presents an efficient method for deducing uncertainties in neural networks. We briefly summarise the main conclusions.

Aleatoric and Epistemic uncertainties provides a nice-to-have mental model of where uncertainties arise, however, the disentanglement is hard and therefore of limited practical value.

Learned versus deduced uncertainties provides a more practical mental model, highlighting that deduced uncertainties are good for out-of-distribution (OOD) whereas learned are more suited for in-distribution (ID) data.

Laplace approximation (LA) is a method to deduce uncertainties. We propose an online training procedure that improves the calibration and stability of the learned weight posterior. Further, we presented an efficient Hessian approximation that allow us to scale the LA to large neural networks with high output dimensions.

CHAPTER 3 Bayesian Image Retrieval

Recall that the first step of the 3D reconstruction pipeline is to retrieve the images useful for the reconstruction. Depending on the object or scene we wish to reconstruct, different meta-data can be used to find the relevant images, e.g. GPS, tags, text descriptions, etc.

In this chapter, we consider the extreme case, where we only have access to images for the retrieval process. Thus, given a query image, we wish to retrieve “similar” images from a large database. “Similar” depends on the application, but could be same bird species, same human face, or the same place. The task is often solved by learning representations that are close when images are “similar”. This can be achieved with metric learning that seeks data representations where similar observations are near and dissimilar ones are far. This elegantly allows for building retrieval systems with simple nearest-neighbor search. Such systems easily cope with a large number of classes and new classes can organically be added without retraining. While these retrieval systems show impressive performance, they quickly, and with no raised alarms, deteriorate with out-of-distribution data [SJ19]. In particular, in safety-critical applications, the lack of uncertainty estimation is a concern as retrieval errors may propagate unnoticed through the system, resulting in erroneous and possibly dangerous decisions. This chapter presents the Mapillary Street-Level Sequences, a large dataset for training retrieval systems, and then extends the Laplace approximation to deduce uncertainties for metric learning systems.

This chapter contains parts from Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition [War+20] and Bayesian Metric Learning for Uncertainty Quantification in Image Retrieval [War+23a]. Not all experiments, results, and related works are presented to keep the text shorter.

3.1 Visual Place Recognition

Visual place recognition consists of recognizing the geographical location of a place based only on image data. The place recognition task is often cast as a retrieval problem. Given a query image, we wish to find the images from the same geographical

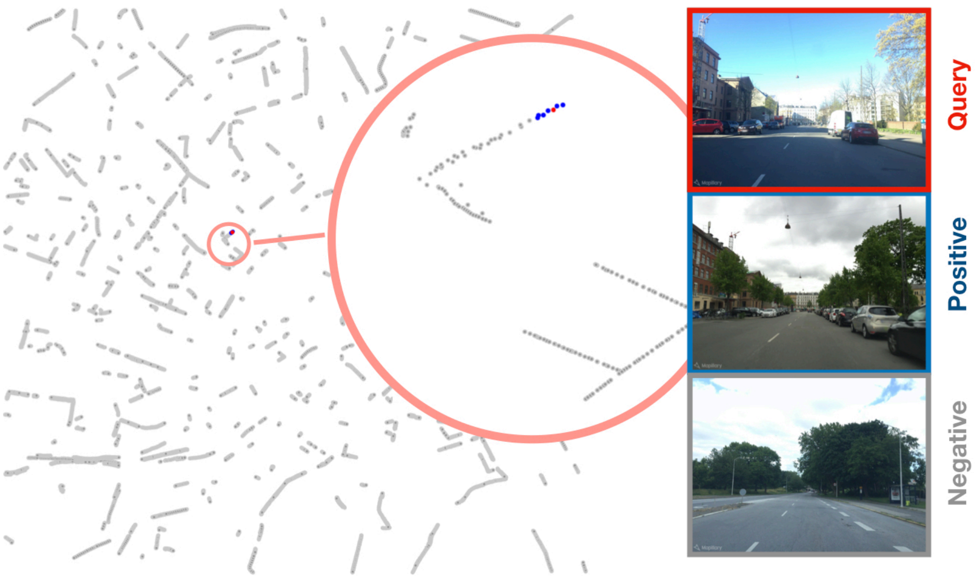


Figure 3.1: Shows a query image that has 6 positive images from the Copenhagen database. In this example, we use a 25 meter as the threshold to define positive images.

location as the query image. Figure 3.1 illustrates the images from the same location (positives) and images from other locations (negatives).

The retrieval problem is solved by building a low dimensional, place-descriptor for both query and database images, and for each query image retrieve the closest n database images [Low+16; Zaf+19]. The goal is to build a robust place descriptor that efficiently represents an exact geographical location based only on image information.

To push the state-of-the-art in lifelong place recognition, we present *Mapillary Street-Level Sequences (MSLS)*, the largest dataset for place recognition to date, with the widest variety of perceptual changes and the broadest geographical spread. MSLS comprises 1.6 million images from Mapillary[1] platform, and covers a wide range of appearance changes, such as different seasons, changing weather conditions, varying illumination at different times of the day, dynamic objects such as moving pedestrians or cars, structural modifications such as roadworks or architectural work, camera intrinsic, and viewpoints as illustrated in Figure 3.2. Figure 3.3 shows that the dataset spans six continents, including diverse cities like Kampala, Zurich, Amman, and Bangkok.

We benchmark a multitude of state-of-the-art models, and find that the state-of-the-art place descriptors, despite impressive performance, fail without any raised alarms, when the query images are of low quality, ambigous or out of distribution

[1] www.mapillary.com

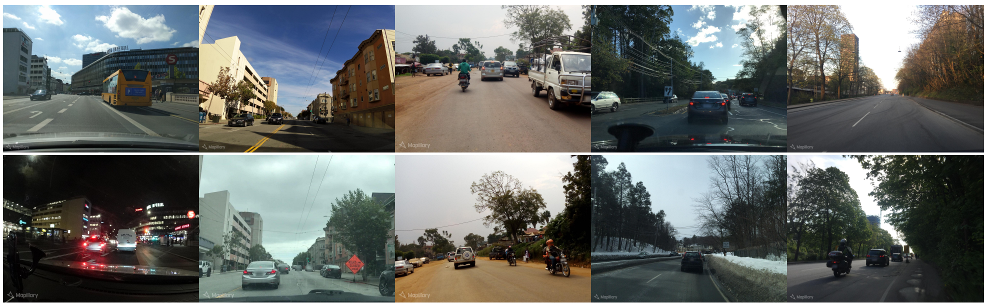


Figure 3.2: Mapillary SLS pairs showing day/night, weather, seasonal, structural, viewpoint, and domain changes. Each column shows images from the same place that are recorded at different dates.

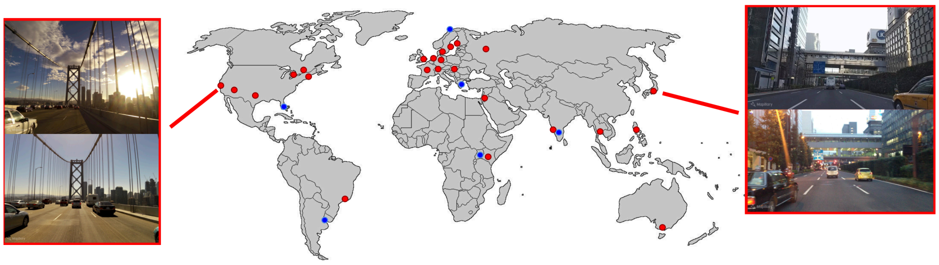


Figure 3.3: Mapillary SLS contains imagery from 30 major cities around the world; red stands for training cities and blue for test cities. See two samples from San Francisco and Tokyo with challenging appearance changes due to dynamic and illumination differences.

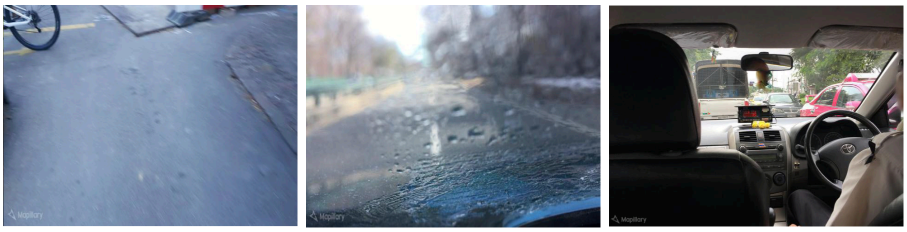


Figure 3.4: **Examples of uncertain queries.** State-of-the-art networks that does model uncertainties, which can lead to silent a failure.

as exemplified in Figure 3.4. In these situations, we would expect the model to associate high uncertainty with its prediction, however, state-of-the-art models provide no notion of uncertainty.

3.2 Stochastic Embeddings

Instead of modelling place-descriptors as low-dimensional deterministic embeddings, we can model them as stochastic ones. This allow us to estimate the uncertainties of queries and retrieved images. We show that the online Laplace Approximation described in the previous chapter can be extended to metric learning and provide reliable stochastic embeddings.

Figure 3.5 illustrate our Bayesian mapping from image to latent space. We estimate the weight posterior of the embedding network f_θ such that we can sample data embeddings to propagate uncertainty through the decision process. The embedding network is parametrized by $\theta \in \Theta$ and trained with the contrastive loss [HS06]. The network maps an image $x \in \mathcal{X} := \mathbb{R}^{HWC}$ to an embedding $z \in \mathcal{Z}$, which is restricted to be on a Z-dimensional unit sphere $\mathcal{Z} := \mathcal{S}^Z$. This spherical normalization is common in retrieval to obtain faster retrieval and a slight performance boost [Ara+16; RTC18].

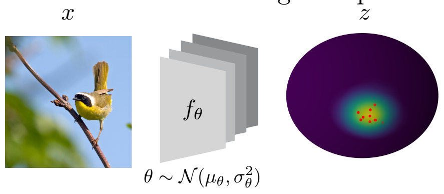


Figure 3.5: **Model overview**. We learn a distribution over parameters, such that we embed an image through sampled encoders f_θ to points z_i (red dots) in a latent space $\mathcal{Z}$. We reduce these latent samples to a single measure of uncertainty by estimating the parameters of a von Mises-Fisher distribution.

The probabilistic contrastive likelihood. The Laplace Approximation (LA) expects the loss function to be a log-posterior, that is log-likelihood plus log-prior. A simple choice of the log-prior is $||\theta||_2^2$ (weight decay). Next, we show that the contrastive loss [HS06] has a probabilistic interpretation and is a negative log-likelihood on the spherical space. Here, we only stretch the proof, and refer to [War+23a] for more details. We define the attractive term $\mathcal{P}^{\rightarrow\leftarrow}(z|x,\theta) \sim \mathcal{N}^S(z|f_\theta(x),\kappa)$ and the repelling term $\mathcal{P}^{\leftarrow\rightarrow}(z|x,\theta) \sim \mathcal{N}^S(z|-f_\theta(x),\kappa)$ as Von Mises-Fisher distributions on the latent spherical space, where the concentration parameter $\kappa > 0$. The product of these two likelihoods yields a contrastive likelihood, which is valid on the spherical space, and its negative log-likelihood is equivalent to the contrastive loss. Thus, we can use the LA with the constrastive loss.

Hessian of the contrastive loss. Both post-hoc and online LA require the Hessian of the contrastive loss $\nabla_\theta^2 \mathcal{L}_{\text{con}}(\theta; \mathcal{D})$. The Hessian is commonly approximated with the Generalized Gauss-Newton (GGN) approximation [FH97; Dax+21; DKH20; Det+21]. The GGN decomposes the loss into $\mathcal{L} = g \circ f$, where g is usually chosen as the loss function and f the model function, and only f is linearized [KHB19].

However, in our case, this decomposition is non-trivial. Recall that the last layer of our network is an ℓ_2 normalization layer, which projects embeddings onto a hypersphere. This normalization layer can either be viewed as part of the model f (linearized normalization layer) or part of the loss g (non-linearized normalization layer).

The former can be interpreted as using the *Euclidean* distance and the latter as using the *Arccos* distance for the contrastive loss. These two share the zero- and first-order terms for normalized embeddings but not the second-order derivatives due to the GGN linearization. The Euclidean interpretation leads to simpler derivatives and interpretations, and we will therefore use it for our derivations. We emphasize that the Arccos is theoretically a more accurate approximation because the ℓ_2-layer is not linearized. The GGN matrix for contrastive loss with the *Euclidean* interpretation is given by

$$\begin{aligned}\nabla^2_\theta \mathcal{L}_{\text{con}}(\theta;\mathcal{I}) &= \sum_{ij\in\mathcal{I}} H^{ij}_\theta = \sum_{ij\in\mathcal{I}_p} H^{ij}_\theta + \sum_{ij\in\mathcal{I}_n} H^{ij}_\theta \\ &\overset{\text{GGN}}{\approx} \sum_{ij\in\mathcal{I}_p} {J^{ij}_\theta}^\top \underbrace{\begin{pmatrix} 1 & -1 \\ -1 & 1\end{pmatrix}}_{:=H_p} J^{ij}_\theta + \sum_{ij\in\mathcal{I}_n} {J^{ij}_\theta}^\top \underbrace{\begin{pmatrix} -1 & 1 \\ 1 & -1\end{pmatrix}}_{:=H_n} J^{ij}_\theta,\end{aligned} \tag{3.1}$$

where $J^{ij}_\theta = \left(J_\theta f_\theta(x_i)^\top, J_\theta f_\theta(x_j)^\top\right)^\top$, with J_θ is the Jacobian wrt. the parameters and H_p and H_n are the Hessians of the contrastive loss wrt. the model output for positive and negative pairs. The first sum runs over positive pairs and the second sum runs over negative pairs *within the margin*. Negative pairs outside the margin do not contribute to the Hessian, and can therefore be ignored to reduce the computational load. The eigenvalues of the Hessian wrt. to the output are $(0, 2)$ and $(-2, 0)$ for the positive H_p and negative H_n terms, so we are not guaranteed to have a positive semidefinite Hessian, H_θ. We propose three solutions to ensure a positive semidefinite Hessian to avoid covariances with negative eigenvalues.

Ensuring positive definite covariance matrix. We do not want to be restricted in the choice of the prior except to have non-zero precision, so we must ensure that $\nabla^2_\theta \mathcal{L}_{\text{con}}(\theta^*;\mathcal{D})$ is positive semidefinite. Differently from the standard convex losses, this is not ensured by the GGN approximation [IKB21]. Our main insight is that we can ensure a positive semidefinite Hessian H_θ by only manipulating the Hessians H_p and H_n in (3.1).

1. Positive: The repelling term is ignored, such that only positive pairs contribute to the Hessian.

$$H_p = \begin{pmatrix} 1 & -1 \\ -1 & 1\end{pmatrix}, \qquad H_n = \begin{pmatrix} 0 & 0 \\ 0 & 0\end{pmatrix} \tag{3.2}$$

2. Fixed: The cross derivatives are ignored.

$$H_p = \begin{pmatrix} 1 & 0 \\ 0 & 1\end{pmatrix}, \qquad H_n = \begin{pmatrix} -1 & 0 \\ 0 & -1\end{pmatrix} \tag{3.3}$$

3. Full: Positive semidefiniteness is ensured with ReLU, $\max(0, \nabla^2_\theta \mathcal{L}_{con}(\theta;\mathcal{D}))$*, on the Hessian of the loss wrt. the parameters.*

$$H_p = \begin{pmatrix} 1 & -1 \\ -1 & 1\end{pmatrix}, \qquad H_n = \begin{pmatrix} -1 & 1 \\ 1 & -1\end{pmatrix} \tag{3.4}$$

The *positive* approximation is inspired by [SJ19], which only uses positive pairs to train an uncertainty module. The gradient arrows in Figure 3.6a illustrate that negative pairs are neglected when computing the Hessian of the contrastive loss. The *fixed* approximation considers one data point at a time, assuming the other one is fixed. Thus, given a pair of data points, this can be interpreted as first moving one data point, and then the second (rather than both at the same time). Figure 3.6b illustrate this idea when all points except a are fixed. Lastly, we propose the *full* Hessian of the contrastive loss (Figure 3.6c) and ensure positive semidefiniteness by computing the ReLU of the diagonal Hessian. This approximation can equivalently be interpreted as a projection into the space of positive semidefinite (PSD) matrices. In practice, the Hessian scales quadratically in memory wrt. the number of parameters. To mitigate this, we approximate this Hessian by its diagonal and only apply the LA on the last layer [LDS89; DL90]. We experimentally find that the *fixed* approximation yields the best performance (Section 3.3).

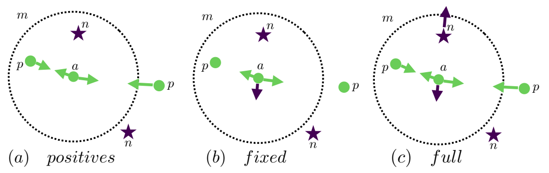


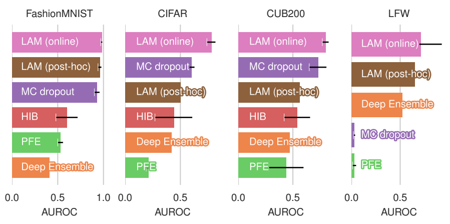
Figure 3.6: **Hessian approximations.** To ensure a positive semidefinite Hessian approximation we propose three approximations. In (a) only the positives p contribute to the Hessian as the negatives n are ignored. In (b) we consider one point at a time, e.g., only the anchor a contributes. In (c) we consider all interactions.

Hard negative mining. Most pairs, namely the negatives outside the margin, do not contribute to the Hessian, so it is wasteful to compute their Hessian. Therefore, we use hard negative mining [MBL20] to only compute the Hessian of pairs that have non-zero Hessian, *i.e.* the negative sample lie within the margin (illustrated with the dotted line in Figure 3.6).

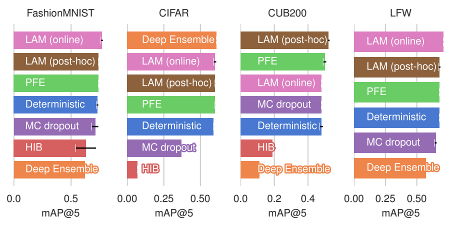
3.3 Experiments

We benchmark our method against strong probabilistic retrieval models. Probabilistic Face Embeddings (PFE) [SJ19] and Hedge Image Embedding (HIB) [Oh+18] perform amortized inference and thus estimate the mean and variance of latent observation. We also compare against MC Dropout [GG16] and Deep Ensemble [LPB17], two approximate Bayesian methods, which have successfully been applied in image retrieval [Tah+19a; Tah+19b].

We compare the models' *predictive performance* with the recall (recall@k) and mean average precision (mAP@k) among the k nearest neighbors [War+21; MBL20; Ara+16]. We evaluate the models' abilities to *interpolate* and *extrapolate* uncertainties by measuring the Area Under the Sparsification Curve (AUSC), Expected Calibration Error (ECE) on in-distribution (ID) data, the Area Under Receiver Operator Curve



((a)) Out-of-distribution detection.



((b)) Predictive performance.

Figure 3.7: **Summary of experimental results.** LAM consistently outperforms existing methods on OoD detection, as measured by AUROC, and matches or surpasses in predictive performance measured by mAP@k. Error bars show one std across five runs.

(AUROC), and Area Under Precision-Recall Curve (AUPRC) on out-of-distribution (OoD) data.

Experimental Summary. We first summarize our experimental results. Across five datasets, three network architectures, and three different sizes of the latent space (ranging from 3 to 2048), we find that LAM has well-calibrated uncertainties, reliably detects OoD examples, and achieves state-of-the-art predictive performance. Figure 3.7(a) shows that the uncertainties from online LAM reliably identify OoD examples. Online LAM outperforms other Bayesian methods, such as post-hoc LAM and MC dropout, on this task, which in turn clearly improves upon amortized methods that rely on a neural network to extrapolate uncertainties. Figure 3.7(b) shows that LAM consistently matches or outperforms existing image retrieval methods in terms of predictive performance. We find that the fixed Hessian approximation with the Arccos distance performs the best, especially on higher dimensional data.

Ablation: Positive definite covariance matrix. We experimentally study which method to ensure a positive semidefinite Hessian has the best performance measured in both predictive performance (mAP@5) and uncertainty quantification (AUROC, AUSC).

We found that all methods perform similarly on simple datasets and low dimensional hyper-spheres, but the fixed approximation with Arccos dis-

Table 3.1: **Ablation on Hessian approximation and GGN decomposition.** Online LA with the fixed approximation and Arccos distance performs best. Error bars show one std. across five runs.

		mAP@5 ↑	AUROC ↑	AUSC ↑
Post-hoc	Euclidean fix	0.70 ± 0.0	0.57 ± 0.25	0.44 ± 0.01
	Euclidean pos	0.70 ± 0.0	0.58 ± 0.23	0.45 ± 0.01
	Euclidean full	0.70 ± 0.0	0.56 ± 0.26	0.44 ± 0.01
	Arccos fix	0.69 ± 0.0	0.53 ± 0.20	0.46 ± 0.02
	Arccos pos	0.70 ± 0.0	0.29 ± 0.11	0.48 ± 0.01
	Arccos full	0.69 ± 0.0	0.55 ± 0.18	0.45 ± 0.01
Online	Euclidean fix	0.63 ± 0.01	0.77 ± 0.04	0.31 ± 0.02
	Euclidean pos	0.70 ± 0.0	0.38 ± 0.10	0.47 ± 0.01
	Euclidean full	0.67 ± 0.01	0.59 ± 0.04	0.42 ± 0.01
	Arccos fix	**0.71 ± 0.0**	**0.78 ± 0.18**	0.50 ± 0.03
	Arccos pos	0.70 ± 0.0	0.23 ± 0.03	0.46 ± 0.00
	Arccos full	**0.71 ± 0.0**	0.70 ± 0.12	**0.51 ± 0.01**

Table 3.2: **Results on MSLS.** LAM yields state-of-the-art uncertainties and matches the predictive performance of deterministic trained models.

	Validation Set						Challenge Set					
	R@1↑	R@5↑	R@10↑	M@5↑	M@10↑	AUSC↑	R@1↑	R@5↑	R@10↑	M@5↑	M@10↑	AUSC↑
Deterministic	**0.77**	**0.88**	**0.90**	**0.61**	**0.56**	—	**0.58**	**0.74**	**0.78**	**0.45**	0.43	—
MC Dropout	0.75	0.87	0.87	0.59	0.54	**0.77**	0.55	0.71	0.76	0.43	0.41	0.57
PFE	**0.77**	**0.88**	**0.90**	**0.61**	**0.56**	0.73	**0.58**	**0.74**	**0.78**	**0.45**	**0.44**	0.57
LAM (post-hoc)	0.76	0.86	0.89	0.60	0.55	0.74	**0.58**	**0.74**	**0.78**	**0.45**	**0.44**	0.59
LAM (online)	0.76	0.87	**0.90**	0.60	**0.56**	**0.77**	0.57	**0.74**	**0.78**	**0.45**	0.43	**0.63**

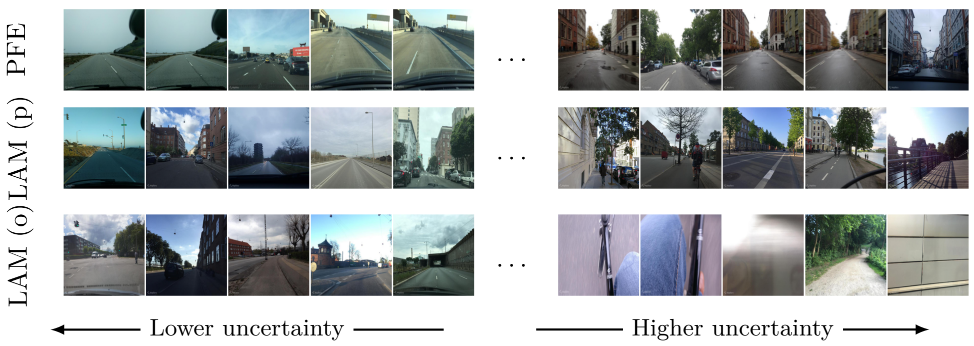


Figure 3.8: **Images with lowest and highest variance** for PFE, post-hoc LAM, and online LAM across MSLS validation set. LAM reliably associates high uncertainty to images that are blurry, are captured facing the pavement, or contain vegetation. These images do not contain features that are descriptive of a specific place, making them especially challenging to geographically locate.

tance performs better on more challenging datasets and higher dimensional hyperspheres.

We present results on one of these more challenging datasets, namely the LFW [Hua+07] face recognition dataset with the CUB200 [Wah+11] bird dataset as an OoD dataset. We use a ResNet50 [He+16] with a GeM pooling layer [RTC18] and a 2048 dimensional embedding and diagonal, last-layer LA [Dax+21].

Table 3.2 shows that online LAM yields state-of-the-art uncertainties for visual place recognition measured with AUSC, while matching the predictive performance of the alternative probabilistic and deterministic methods on both the MSLS validation and the challenge set. Figure 3.9 shows the sparsification curves on the challenge set. Both

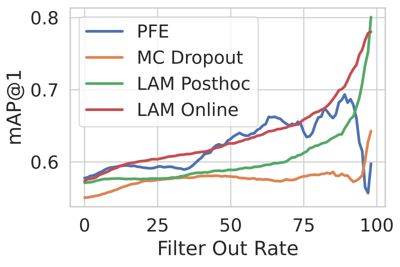


Figure 3.9: **Sparsification curve.** Online and post-hoc LAM's sparsification curves monotonically increase, showing that they reliably associate higher uncertainty with harder observations.

online and post-hoc LAM have monotonically increasing sparsification curves, implying that when we remove the most uncertain observations, the predictive performance increase. This illustrates that LAM produces reliable uncertainties for this challenging open-set retrieval task. Figure 3.8 shows the queries associated with the highest and lowest uncertainty. LAM predicts high uncertainty to images that are blurry, captured facing into the pavement, or contain mostly vegetation. These images do not have features that are descriptive of a specific place, making them hard to geographically locate. Assigning high uncertainty to these images greatly reduces the risk of silent failures during the retrieval process.

3.4 Summary

In this chapter, we explored image retrieval systems to recover images for our reconstruction. The main conclusions are as follows:

Mapillary SLS. We presented a large place recognition dataset containing a large array of visual appearance changes, such as day-night, view-point, structural, and dynamic.

Silent failure. Existing state-of-the-art retrieval models do not assign uncertainties to their predictions, which might lead to silent failures.

Stochastic Embeddings. The Laplace approximation can be adapted to metric learning and can be used to deduce reliable uncertainty estimates for latent representations while matching the predictive performance of state-of-the-art systems. These uncertainties reduce the risk of silent failures.

CHAPTER 4

Uncertainties in Visual Localization

The second step of the 3D reconstruction pipeline is to estimate the camera location for each image. This is commonly done by simultaneously building a sparse 3D model via keypoint matching and optimizing for the camera location [HZ04]. This is referred to as structure from motion (SfM) and has been studied for decades in the computer vision and robotics community [Özy+17; HZ04]. Recently, deep learning-based keypoint detectors have demonstrated better performance than handcrafted detectors, especially being more robust to appearance variations, e.g. day-night changes [Sar+19; Bar+22a; Zho+22; Xu+22]. However, the detected features a predicted without any notion of uncertainties, leading to suboptimal performance for camera localization. In this chapter, we explore a method to deduce uncertainties for any state-of-the-art deep detector.

This chapter contains parts from DAC: Detector-Agnostic Spatial Covariances for Deep Local Features [TWC23]. Not all experiments, results, and related works are presented to keep the text shorter.

4.1 Detector-Agnostic Covariances Estimates

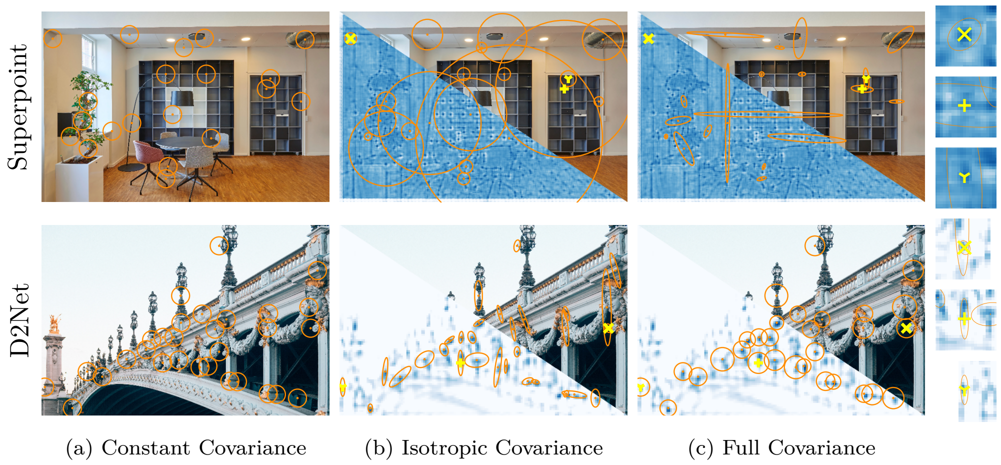


Figure 4.1: **Detector-agnostic, post-hoc uncertainty for learned local-feature detectors.** State-of-the-art deep feature detectors, such as Superpoint [DMR18] and D2Net [Dus+19], do not estimate the spatial uncertainty of their detections. This corresponds to assuming constant covariances (a), which leads to suboptimal performance. We propose two methods to model local features' uncertainty: (b) a point-wise isotropic covariance and (c) a structure-based full covariance estimate. We demonstrate that modeling uncertainties lead to improved performance in downstream tasks such as solving the perspective-n-point (PnP) problem and nonlinear optimizations.

Deep learned detectors and descriptors of local features (keypoints) are more robust to viewpoint and appearance changes than handcrafted ones [Sar+19; Bar+22a; Zho+22; Xu+22]. However, these learned systems lack uncertainty modeling. This absence of probabilistic formulation implicitly assumes *constant* spatial covariances (see Figure 4.1a) for downstream geometric estimation tasks, leading to suboptimal results.

We propose to model the spatial covariances of learned keypoints. We leverage that learned detectors share a common design, where a CNN predicts a score map from which keypoints are extracted. We propose two post-hoc covariance matrix estimates of each keypoint location: (1) an isotropic estimator that directly uses the regressed score, and (2) a theoretically-motivated estimator of the full spatial covariance using the local structure tensor. Figure 4.1b-c shows exemplar keypoints and their estimated covariances overlaid on their corresponding images and score maps.

4.2 Method

Our framework takes as input a $H \times W$ RGB image $\mathbf{I}$, and outputs local features with their *spatial* uncertainties. It is composed of (1) a pretrained feature detector and (2) our novel detector-agnostic uncertainty module (Figure 4.2).

4.2.1 Overview

Pre-trained local-feature detector. Our framework is applicable to the vast majority of learned detectors [Ver+15; Yi+16; DMR18; Rev+19; Dus+19; Luo+20; TFT20; BM22]. These share a common design, using CNNs to predict a score map $\mathbf{S} \in \mathbb{R}^{kH \times kW}$ followed by non-maximum suppression (NMS) to extract a sparse set of local features. We leverage this standard design to make a detector-agnostic, post-hoc uncertainty module that takes $\mathbf{S}$ as input and estimates the spatial uncertainty of each detected local feature.

Detector-Agnostic Feature Uncertainties. More formally, we consider that the location $\mathbf{x}_i \in \mathbb{R}^2$ of a local feature i, detected in $\mathbf{I}$, stems from perturbing its *true* location $\mathbf{x}_i = \mathbf{x}_{i,\text{true}} + \boldsymbol{\xi}_i$, with random noise $\boldsymbol{\xi}_i \in \mathbb{R}^2$ whose distribution we want to estimate. We follow the dominant model in computer vision [Kan96; Tri+00; HZ04], which uses *second order statistics* to describe the uncertainty:

$$\mathbb{E}[\boldsymbol{\xi}_i] = \mathbf{0} \ , \quad \boldsymbol{\Sigma}_i := \mathrm{Cov}(\boldsymbol{\xi}_i) = \mathbb{E}[\boldsymbol{\xi}_i \boldsymbol{\xi}_i^\top] \ , \quad \forall i \ . \tag{4.1}$$

Thus, our goal is to quantify each *covariance matrix* $\boldsymbol{\Sigma}_i$. To this end, we propose two methods: (1) A point-wise and isotropic estimator (Section 4.2.2), and (2) a *full* covariance estimator based on the local structure tensor (Section 4.2.3). Intuitively, a *peaky* learned pattern around a detection will yield low spatial uncertainty, whereas a flatter pattern will yield larger spatial uncertainty.

4.2.2 Point-wise Estimation

Our first estimator of the spatial uncertainty uses the regressed score of each keypoint

$$\boldsymbol{\Sigma}_i := \frac{1}{\mathbf{S}(\mathbf{x}_i)} \, \mathbf{I}_{2\times 2} = \begin{bmatrix} 1/\mathbf{S}(\mathbf{x}_i) & 0 \\ 0 & 1/\mathbf{S}(\mathbf{x}_i) \end{bmatrix} \ . \tag{4.2}$$

Figure 4.1 shows that this estimator yields isotropic predictions of uncertainty (equal in all directions), so it only quantifies the relative scale regardless of the learned local structure.

4.2.3 Structure Tensor

Quantification of local saliency motivates the use of the local structure tensor, $\mathbf{C}_i \in \mathbb{R}^{2\times 2}$ [HS+88]. Defining $[\nabla_x S_i, \nabla_y S_i] := \partial \mathbf{S}/\partial \mathbf{x}|_{\mathbf{x}_i}$ as the spatial gradient of $\mathbf{S}$ evalu-

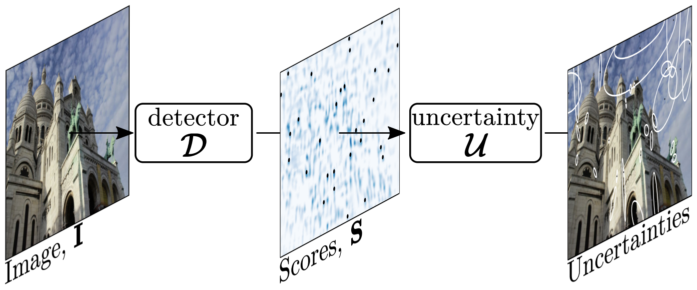


Figure 4.2: **Method overview**. We exploit the dominant approach in *learned* local-feature detectors, represented by $\mathcal{D}$, of regressing for each input image $\mathbf{I}$, a map of scores $\mathbf{S} := \mathcal{D}(\mathbf{I})$, over which detections are done. We propose to quantify the uncertainty of each detected location $\mathbf{x}_i$ by a mapping $\mathcal{U}(\mathbf{S}, \mathbf{x}_i)$ agnostic to $\mathcal{D}$.

ated at $\mathbf{x}_i$, $\mathbf{C}_i$ in its local window $\mathcal{W}_i$ is given by

$$\mathbf{C}_i := \sum_{j \in \mathcal{W}_i} w_j \left.\frac{\partial \mathbf{S}}{\partial \mathbf{x}}\right|_{\mathbf{x}_j}^{\top} \left.\frac{\partial \mathbf{S}}{\partial \mathbf{x}}\right|_{\mathbf{x}_j} = \sum_{j \in \mathcal{W}_i} w_j \begin{bmatrix} (\nabla_x S_j)^2 & \nabla_x S_j \nabla_y S_j \\ \nabla_y S_j \nabla_x S_j & (\nabla_y S_j)^2 \end{bmatrix} , \tag{4.3}$$

with $w_j \in \mathbb{R}^+$ being the weight of pixel j, preferably defined by a Gaussian centered at $\mathbf{x}_i$ [SMS18].

Statistical interpretation. Under a Gaussian model of aleatoric uncertainty, common in deep learning [KG17; GDS20] and motivated by the principle of maximum entropy [MBF09] of the local features' scores, we can set up a parametric optimization of $\mathbf{t} \in \mathbb{R}^2$ which maximizes the *likelihood*, $\mathcal{L}(\mathbf{t}_i \mid \mathcal{S})$ with $\mathcal{S} := \{\underline{\mathbf{S}}(\mathbf{x}_j) \mid \mathbf{x}_j \in \mathcal{W}_i\}$, of the observations:

$$\underline{\mathbf{S}}(\mathbf{x}_i) = \mathbf{S}(\mathbf{x}_i + \mathbf{t}_i) + \varepsilon_i \ , \quad \text{with} \quad \varepsilon_i \sim \mathcal{N}(0,\ \sigma^2) \ , \tag{4.4}$$

where $\underline{\mathbf{S}}(\mathbf{x}_i)$ is the observed score, perturbed by the random noise ε_i, independently affecting the rest of observations. Thus, the optimization is formulated as follows

$$\hat{\mathbf{t}}_i = \arg\max_{\mathbf{t}_i} \prod_{j \in \mathcal{W}_i} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\underline{\mathbf{S}}(\mathbf{x}_j) - \mathbf{S}(\mathbf{x}_j + \mathbf{t}_i))^2}{2\sigma^2}\right) . \tag{4.5}$$

Its solution, or *maximum-likelihood-estimation* (MLE), is known *a priori*: $\hat{\mathbf{t}}_i = \mathbf{0}$, which is *unbiased* given our statistical model (4.4), and coherent with (4.1).

With these conditions, the inverse of the *Fisher information matrix*, $\boldsymbol{\mathcal{I}}(\hat{\mathbf{t}})$, evaluated at the MLE, imposes a lower bound in the covariance matrix of the estimator $\hat{\mathbf{t}}_i = \mathbf{0}$, known as *Cramer-Rao Lower Bound* (CRLB) [HS+07]:

$$\mathrm{Var}(\hat{\mathbf{t}}_i) \geq \boldsymbol{\mathcal{I}}(\hat{\mathbf{t}}_i)^{-1} \ . \tag{4.6}$$

$\boldsymbol{\mathcal{I}}(\hat{\mathbf{t}})$ is defined as $\mathbb{E}[\mathbf{s}_i^\top \mathbf{s}_i]$, being the variance of the log-likelihood derivative (known as *score*) $\mathbf{s}_i \coloneqq \partial \log \mathcal{L}(\mathbf{t}_i \mid \mathcal{S})/\partial \mathbf{t}_i$, since $\mathbb{E}[\mathbf{s}_i] = \mathbf{0}$ [HS+07]. In our case, it is given by

$$\mathbf{s}_i = \sum_{\mathbf{x}_j \in \mathcal{W}_i} \left(\mathbf{d}_j \coloneqq \frac{\underline{\mathbf{S}}(\mathbf{x}_j) - \mathbf{S}(\mathbf{x}_j + \mathbf{t}_i)}{\sigma^2} \frac{\partial \mathbf{S}(\mathbf{x}_j + \mathbf{t}_i)}{\partial \mathbf{t}_i} \right) . \tag{4.7}$$

Due to the linearity of expectation and the independence of observations, $\mathbb{E}[\mathbf{d}_j^\top \mathbf{d}_k] = \mathbf{0}$, $\forall j \neq k$. Thereby

$$\boldsymbol{\mathcal{I}}(\hat{\mathbf{t}}_i) = \sum_{j \in \mathcal{W}_i} \mathbb{E}[\mathbf{d}_j^\top \mathbf{d}_j]\big|_{\hat{\mathbf{t}}_i} . \tag{4.8}$$

Derivatives of (4.7) are applied to our deterministic model, thus they can go out of the expectation. Evaluating them on $\hat{\mathbf{t}}_i = \mathbf{0}$ leads to

$$\begin{aligned} \boldsymbol{\mathcal{I}}(\hat{\mathbf{t}}_i) = \sum_{\mathbf{x}_j \in \mathcal{W}_i} \frac{1}{\sigma^4} \left.\frac{\partial \mathbf{S}(\mathbf{x}_j)}{\partial \mathbf{x}}\right|_{\mathbf{x}_j}^\top \left.\frac{\partial \mathbf{S}(\mathbf{x}_j)}{\partial \mathbf{x}}\right|_{\mathbf{x}_j} \\ \mathbb{E}[(\mathbf{S}(\mathbf{x}_j) - \mathbf{S}(\mathbf{x}_j + \mathbf{t}_i))^2]\big|_{\hat{\mathbf{t}}_i} . \end{aligned} \tag{4.9}$$

Lastly, $\mathbb{E}[(\mathbf{S}(\mathbf{x}_j) - \mathbf{S}(\mathbf{x}_j + \mathbf{t}_i))^2]\big|_{\hat{\mathbf{t}}_i} = \mathbb{E}[\varepsilon_i^2] = \sigma^2$ since $\mathbb{E}[\varepsilon_i] = 0$, implying that our Fisher information matrix is

$$\boldsymbol{\mathcal{I}}(\hat{\mathbf{t}}_i) = \frac{1}{\sigma^2} \sum_{\mathbf{x}_j \in \mathcal{W}_i} \left.\frac{\partial \mathbf{S}(\mathbf{x}_j)}{\partial \mathbf{x}}\right|_{\mathbf{x}_j}^\top \left.\frac{\partial \mathbf{S}(\mathbf{x}_j)}{\partial \mathbf{x}}\right|_{\mathbf{x}_j} , \tag{4.10}$$

at the MLE, thus matching the local structure tensor (4.3) up to a scale factor related to $\mathrm{Var}(\varepsilon_i) = \sigma^2$, unknown *a priori*. Recalling the CRLB, although achievable only asymptotically [Tri+00; HS+07], it motivates $\boldsymbol{\Sigma}_i \coloneqq \mathbf{C}_i^{-1}$ as an up-to-scale covariance matrix of each location $\mathbf{x}_i$.

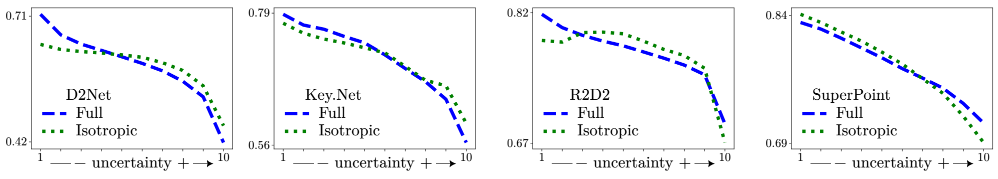


Figure 4.3: **Matching accuracy vs uncertainty on HPatches [Bal+17].** For each detector, we show the averaged MMA across all thresholds. This is done for 10 uniform ranges of uncertainty, ordered from lowest to highest (x-axis). Our covariances correctly model less accurate matches, being the *full*-based approach (structure tensor) the one showing greater sensitivity in D2Net, Key.Net, and R2D2.

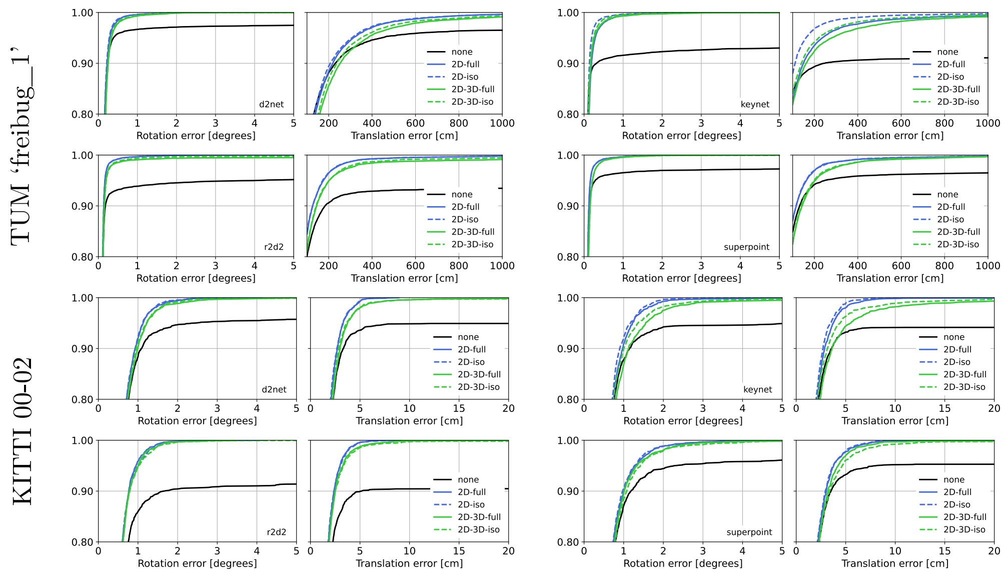


Figure 4.4: **Evaluation in TUM-RGBD [Stu+12] and KITTI [Gei+13].** We report the cumulative errors for the rotation and translation of the camera poses. Practically all estimations converge to acceptable thresholds when leveraging our *2D full* and *2D iso*tropic covariances. This also occurs when using 3D covariances stemming from our 2D ones. Without our covariances, a significant percentage of poses do not converge.

4.3 Experiments

Implementation details. In our experiments, we compute the structure tensor independently of the learned detector: for each local feature i, spatial differentiation at $\mathbf{S}(\mathbf{x}_j)$, $\forall j \in \mathcal{W}_i$, is done with Sobel filters. Integration in $\mathcal{W}_i$ is done with a 7×7 window, the result of using an isotropic Gaussian filter with $\sigma = 1$ of cutoff frequency 3σ. Throughout all the experiments, we evaluate and extract our covariances using the state-of-the-art learned detectors: Key.Net [BM22], Superpoint [DMR18], D2Net [Dus+19] and R2D2 [Rev+19].

4.3.1 Matching accuracy.

We first test the relation between our covariances with the accuracy of local-feature matching. Intuitively, local features detected with higher uncertainty should relate to less accurate matches, and vice versa. For this purpose, we consider the widely adopted HPatches dataset [Bal+17].

Evaluation protocol. We base our evaluation on the one of [Dus+19]. First, extraction of local features and, in our case, covariance matrices of their locations is performed for all images. In all sequences, pairwise matching is done between a reference image r and each remaining image i with Mutual Nearest Neighbor (MNN). We then compute the reprojection errors and their covariances with the homographies, $\mathbf{H}_{i,r}$, provided by the dataset:

$$\mathbf{e}_{i,r} \coloneqq \operatorname{cart}(\hat{\mathbf{x}}_i - \mathbf{H}_{i,r}\hat{\mathbf{x}}_r) \ , \quad \boldsymbol{\Sigma}_{\mathbf{e}_{i,r}} \coloneqq \mathbf{J}\boldsymbol{\Sigma}_{\mathbf{x}_r}\mathbf{J}^\top + \boldsymbol{\Sigma}_{\mathbf{x}_i} \ , \tag{4.11}$$

where cart maps from homogeneous to Cartesian space, and $\mathbf{J} \coloneqq \partial\mathbf{e}_{i,r}/\partial\mathbf{x}_r$, *i.e.* we linearly propagate each $\boldsymbol{\Sigma}_{\mathbf{x}_r}$.

We quantify the uncertainty of the match with the biggest eigenvalue of the corresponding $\boldsymbol{\Sigma}_{\mathbf{e}_{i,r}}$. Based on them, all matches are distributed in 10 uniform ranges from lowest to highest uncertainty. To quantify the accuracy in matching at each range, we use the mean matching accuracy error (MMA—average percentage of matches with a corresponding value of $\|\mathbf{e}_{i,r}\|$ below a threshold). We use the thresholds of [Dus+19]. Finally, we compute the mean of all the MMA values at each range (dubbed $\overline{\text{MMA}}$).

Results. Figure 4.3 shows $\overline{\text{MMA}}$ at each uncertainty range. Ranges are ordered from lowest (1) to highest (10) uncertainty. As can be seen, it exists a direct relation between accuracy and both *full* and *isotropic* covariance estimates. With full covariances, lower uncertainty estimates imply higher accuracy. However, this is not always the case when using isotropic covariances (see R2D2). Additionally, $\overline{\text{MMA}}$ presents higher sensitivity to the uncertainty stemming from full covariances on D2Net, Key.Net, and R2D2.

4.3.2 Geometry estimation

To test the influence of our covariances in 3D-geometry estimation, we follow the evaluation of [Vak+21]. It evaluates common stages in geometric estimation pipelines such as solving the perspective-n-point problem and motion-only bundle adjustment (MO-BA). The data used consists in the sequences 00-02 of KITTI [Gei+13] and the first three 'freiburg_1' RGB sequences of TUM-RGBD [Stu+12].

Evaluation protocol. KITTI is used with a temporal window of two left frames, and three in TUM RGB-D (with a relative distance > 2.5 cm). For each frame, local features and our 2D covariances are extracted. Pairwise matching is done with MNN. To form feature tracks (set of 2D points corresponding to the same 3D point) we use the track separation algorithm of [DSP20]. Matched features are triangulated with GT camera poses and DLT algorithm [HZ04], and refined with *2D-covariance-weighted* (when using uncertainty) Levenberg-Marquardt (LM), producing also covariances for 3D point coordinates. The next frame is used for evaluation. After matching it to the reference images we obtain 2D-3D matches, which are used to localize the new frame. When using no uncertainty, EPnP [LMF09] is chosen as the PnP solver.

Otherwise, when leveraging our proposed 2D covariances, and optionally, the 3D covariances from LM, EPnPU [Vak+21]. Finally, the estimated camera pose is refined with a *covariance-weighted* (when using uncertainty) MO-BA. For more details, we refer the reader to [Vak+21].

To quantify the accuracy of the estimated camera poses, we use the absolute rotation error $e_{\text{rot}} = \arccos(0.5\,\text{trace}(\mathbf{R}_{\text{true}}^{\top}\mathbf{R} - 1))$, and the absolute translation error $e_{\text{t}} = \|\mathbf{t}_{\text{true}} - \mathbf{t}\|$, where $\mathbf{R}_{\text{true}}, \mathbf{t}_{\text{true}}$ is the GT pose and $\mathbf{R}, \mathbf{t}$ is the estimated one.

Results. Figure 4.4 shows the cumulative error curves for each sequence. Practically all pose estimations obtained with methods leveraging our covariances, fall under acceptable error thresholds, whereas the ones from the baseline do not.

4.4 Summary

We present detector-agnostic models for the spatial covariances of deep local features. Specifically, we proposed two methods based on their learned score maps: one using local-feature scores directly, and another theoretically-motivated method using local structure tensors. Our experiments on KITTI and TUM show that our covariances are well calibrated, significantly benefiting 3D-geometry estimation tasks, such as estimating the camera location.

CHAPTER 5

Generative Models for Multiview Stereo

The previous chapter presented a method to deduce uncertainties for local keypoints and showed that these uncertainties can improve camera localization, thus providing a collection of posed images. In this chapter, we present methods to build a dense 3D reconstruction from this collection of posed images. This process is referred to as Multiview Stereo (MVS) and is a long-standing problem in computer vision [FH+15; HZ04]. We focus on Neural Radiance Fields (NeRFs)[Mil+20], which have recently shown impressive capabilities for 3D reconstructing. A NeRF maps a 3D point and view-direction (x, y, z, ϕ, θ) to a color and density value (RGB, σ). It is optimized with volume rendering [Max95; Mil+20], minimizing the reconstruction loss between rendered rays and captured pixels. This allows for novel-view synthesis and has easied the 3D reconstruction process significantly. However, NeRFs are still especially challenged by dynamic scenes. Furthermore, since it is cast as a per-scene optimization problem, NeRFs do not learn reusable 3D priors. This chapter, presents K-planes, an efficient method to learn dynamic NeRFs, and Nerfbusters, a method to incorporate learned 3D generative priors into the reconstruction process. These works aim to enhance the quality of dynamic and in-the-wild 3D reconstructions.

This chapter contains parts from Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs [War+23b] and K-Planes: Explicit Radiance Fields in Space, Time, and Appearance [Fri+23]. Not all experiments, results, and related works are presented to keep the text shorter.

5.1 Dynamic 3D reconstructions

Recent interest in dynamic radiance fields demands representations of 4D volumes. However, storing a 4D volume directly is prohibitively expensive due to the curse of dimensionality. Several approaches have been proposed to factorize 3D volumes for static radiance fields, but these do not easily extend to higher dimensional volumes.

We propose a factorization of 4D volumes that is simple, interpretable, compact, and yields fast training and rendering. Specifically, we use six planes to represent a 4D volume, where the first three represent space and the last three represent space-time

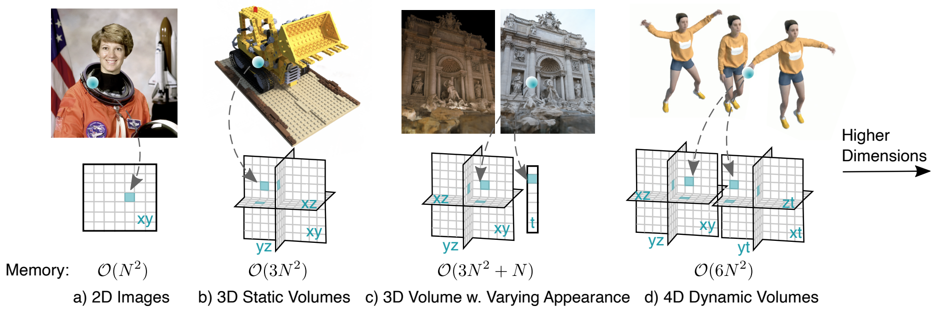

Higher Dimensions
xy
xz
yz
t
zt
xt
yt
Memory: $\mathcal{O}(N^2)$ $\mathcal{O}(3N^2)$ $\mathcal{O}(3N^2+N)$ $\mathcal{O}(6N^2)$
a) 2D Images
b) 3D Static Volumes
c) 3D Volume w. Varying Appearance
d) 4D Dynamic Volumes

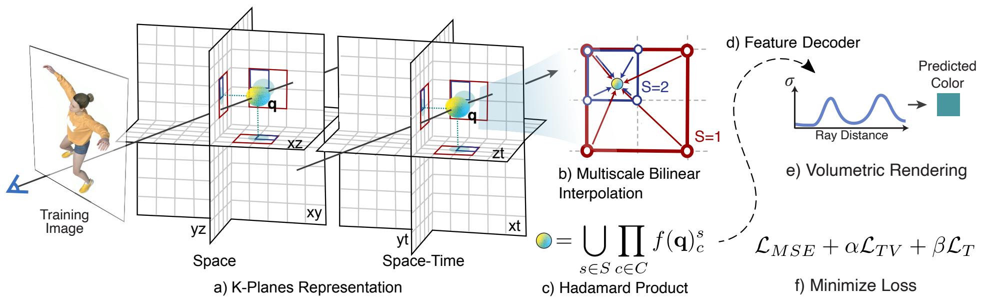
Figure 5.1: **Planar factorization of d-dimensional spaces.** We propose a simple planar factorization for volumetric rendering that naturally extends to arbitrary-dimensional spaces, and that scales gracefully with dimension in both optimization time and model size. We show the advantages of our approach on 3D static volumes, 3D photo collections with varying appearances, and 4D dynamic videos.

changes, as illustrated in Figure 5.1(d). This decomposition of space and space-time makes our model interpretable, i.e. dynamic objects are visible in the space-time planes, whereas static objects only appear in the space planes. This interpretability enables dimension-specific priors in time and space.

More generally, our approach yields a straightforward, prescriptive way to select a factorization of any dimension with 2D planes. For a d-dimensional space, we use $k = \binom{d}{2}$ ("*d-choose*-2") *k-planes*, which represent every pair of dimensions — for example, our model uses $\binom{4}{2} = 6$ *hex-planes* in 4D and reduces to $\binom{3}{2} = 3$ *tri-planes* in 3D. Choosing any other set of planes would entail either using more than k planes and thus occupying unnecessary memory, or using fewer planes and thereby forfeiting the ability to represent some potential interaction between two of the d dimensions. We call our model K-planes; Figure 5.1 illustrates its natural application to both static and dynamic scenes.

Most radiance field models entail some black-box components with their use of MLPs. Instead, we seek a simple model whose functioning can be inspected and understood. We find two design choices to be fundamental in allowing K-planes to be a white-box model while maintaining reconstruction quality competitive with or better than previous black-box models [Li+22a; Pum+21]: (1) Features from our K-planes are *multiplied* together rather than added, as was done in prior work [Cha+22; Che+22], and (2) our linear feature decoder uses a learned basis for view-dependent color, enabling greater adaptivity including the ability to model scenes with variable appearance. We show that an MLP decoder can be replaced with this linear feature decoder only when the planes are multiplied, suggesting that the former is involved in both view-dependent color and determining spatial structure.

Our factorization of 4D volumes into 2D planes leads to a high compression level without relying on MLPs, using 200 MB to represent a 4D volume whose direct representation at the same resolution would require more than 300 GB, a compression



Training Image
yz
xy
xz
q
Space
yt
xt
zt
Space-Time
a) K-Planes Representation
S=2
S=1
b) Multiscale Bilinear Interpolation
$= \bigcup_{s \in S} \prod_{c \in C} f(\mathbf{q})_c^s$
c) Hadamard Product
d) Feature Decoder
σ
Ray Distance
Predicted Color
e) Volumetric Rendering
$\mathcal{L}_{MSE} + \alpha\mathcal{L}_{TV} + \beta\mathcal{L}_T$
f) Minimize Loss

Figure 5.2: **Method overview.** (a) Our K-planes representation factorizes 4D dynamic volumes into six planes, three for space and three for spatiotemporal variations. To obtain the value of a 4D point $\mathbf{q} = (x, y, z, t)$, we first project the point into each plane, in which we (b) do multiscale bilinear interpolation. (c) The interpolated values are multiplied and then concatenated over the S scales. (d) These features are decoded either with a small MLP or our explicit linear decoder. (e) We follow the standard volumetric rendering formula to predict ray color and density. The model is optimized by (f) minimizing the reconstruction loss with simple regularization in space and time.

rate of three orders of magnitude. Furthermore, despite not using any custom CUDA kernels, K-planes trains orders of magnitude faster than prior implicit models and on par with concurrent hybrid models.

In summary, we present the first white-box, interpretable model capable of representing radiance fields in arbitrary dimensions, including static scenes, dynamic scenes, and scenes with variable appearance. Our K-planes model achieves competitive performance across reconstruction quality, model size, and optimization time across these varied tasks, without any custom CUDA kernels. In the following section, we describe the 4D instantiation of our K-planes factorization.

5.2 K-planes model

5.2.1 Hex-planes

The hex-planes factorization uses six planes. We refer to the space-only planes as $\mathbf{P}_{xy}$, $\mathbf{P}_{xz}$, and $\mathbf{P}_{yz}$, and the space-time planes as $\mathbf{P}_{xt}$, $\mathbf{P}_{yt}$, and $\mathbf{P}_{zt}$. Assuming symmetric spatial and temporal resolution N for simplicity of illustration, each of these planes has shape $N\mathbf{x}N\mathbf{x}M$, where M is the size of stored features that capture the density and view-dependent color of the scene.

We obtain the features of a 4D coordinate $\boldsymbol{q} = (i, j, k, \tau)$ by normalizing its entries between $[0, N)$ and projecting it onto these six planes

$$f(\boldsymbol{q})_c = \psi\big(\mathbf{P}_c, \pi_c(\boldsymbol{q})\big), \tag{5.1}$$

where π_c projects $\boldsymbol{q}$ onto the c'th plane and ψ denotes bilinear interpolation of a point into a regularly spaced 2D grid. We repeat (5.1) for each plane $c \in C$ to obtain feature vectors $f(\boldsymbol{q})_c$. We combine these features over the six planes using the Hadamard product (elementwise multiplication) to produce a final feature vector of length M

$$f(\boldsymbol{q}) = \prod_{c \in C} f(\boldsymbol{q})_c. \tag{5.2}$$

These features will be decoded into color and density using either a linear decoder or an MLP, described in Section 5.2.3.

Why Hadamard product? In 3D, K-planes reduces to the tri-plane factorization, which is similar to [Cha+22] except that the elements are multiplied. A natural question is why we multiply rather than add, as has been used in prior work with tri-plane models [Cha+22; Pen+20]. Figure 5.3 illustrates that combining the planes by multiplication allows K-planes to produce spatially localized signals, which is not possible with addition.

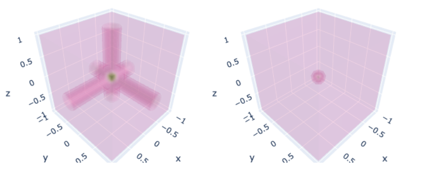


Figure 5.3: **Addition versus Hadamard product.** Elementwise addition of plane features (left) compared to multiplication (right), in a triplane example. A single entry in each plane is positive and the rest are zero, selecting a single 3D point by multiplication but producing intersecting lines by addition. This selection ability of multiplication improves the expressivity of our explicit model.

This selection ability of the Hadamard product produces substantial rendering improvements for linear decoders and modest improvement for MLP decoders, as shown in Table 5.1. This suggests that the MLP decoder is involved in both view-dependent color and determining spatial structure. The Hadamard product relieves the feature decoder of this extra task and makes it possible to reach similar performance using a linear decoder solely responsible for view-dependent color.

Plane Combination	Explicit	Hybrid	# params ↓
Multiplication	35.29	35.75	33M
Addition	28.78	34.80	33M

Table 5.1: **Ablation study over Hadamard product.** Multiplication of plane features yields a large improvement in PSNR ↑ for our explicit model, whereas our hybrid model can use its MLP decoder to partially compensate for the less expressive addition of planes. This experiment uses the static *Lego* scene [Mil+20] with 3 scales: 128, 256, and 512, and 32 features per scale.

5.2.2 Interpretability

The separation of space-only and space-time planes makes the model interpretable and enables us to incorporate dimension-specific priors. For example, if a region of the scene never moves, its temporal component will always be 1 (the multiplicative identity), thereby just using the features from the space planes. This offers compression benefits since a static region can easily be identified and compactly represented. Furthermore, the space-time separation improves interpretability, i.e. we can track the changes in time by visualizing the elements in the time-space planes that are not 1. This simplicity, separation, and interpretability make adding priors straightforward.

Multiscale planes. To encourage spatial smoothness and coherence, our model contains multiple copies at different spatial resolutions, for example 64, 128, 256, and 512. Models at each scale are treated separately, and the M-dimensional feature vectors from different scales are concatenated together before being passed to the decoder. The red and blue squares in Figure 5.2a-b illustrate bilinear interpolation with multiscale planes. Inspired by the multiscale hash mapping of Instant-NGP[Mül+22], this representation efficiently encodes spatial features at different scales, allowing us to reduce the number of features stored at the highest resolution and thereby further compressing our model. Empirically, we do not find it necessary to represent our time dimension at multiple scales.

Total variation in space. Spatial total variation regularization encourages sparse gradients (with L1 norm) or smooth gradients (with L2 norm), encoding priors over edges being either sparse or smooth in space. We encourage this in 1D over the spatial dimensions of each of our space-time planes and in 2D over our space-only planes:

$$\mathcal{L}_{TV}(\mathbf{P}) = \frac{1}{|C|n^2} \sum_{c,i,j} \left(\|\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i-1,j}\|_2^2 + \|\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i,j-1}\|_2^2\right), \tag{5.3}$$

where i, j are indices on the plane's resolution. Total variation is a common regularizer in inverse problems and was used in Plenoxels [Fri+22] and TensoRF [Che+22]. We use the L2 version in our results, though we find that either L2 or L1 produces similar quality.

Smoothness in time. We encourage smooth motion with a 1D Laplacian (second derivative) filter

$$\mathcal{L}_{smooth}(\mathbf{P}) = \frac{1}{|C|n^2} \sum_{c,i,t} \|\mathbf{P}_c^{i,t-1} - 2\mathbf{P}_c^{i,t} + \mathbf{P}_c^{i,t+1}\|_2^2, \tag{5.4}$$

to penalize sharp "acceleration" over time. We only apply this regularizer on the time dimension of our space-time planes.

Sparse transients. We want the static part of the scene to be modeled by the space-only planes. We encourage this separation of space and time by initializing the features in the space-time planes as 1 (the multiplicative identity) and using an ℓ_1 regularizer on these planes during training:

$$\mathcal{L}_{sep}(\mathbf{P}) = \sum_c \|\mathbf{1} - \mathbf{P}_c\|_1, \qquad c \in \{xt, yt, zt\}. \tag{5.5}$$

In this way, the space-time plane features of the K-planes decomposition will remain fixed at 1 if the corresponding spatial content does not change over time.

5.2.3 Feature decoders

We offer two methods to decode the M-dimensional temporally- and spatially-localized feature vector $f(\boldsymbol{q})$ from (5.2) into density, σ, and view-dependent color, $\boldsymbol{c}$.

Learned color basis: a linear decoder and explicit model. Plenoxels [Fri+22], Plenoctrees [Yu+21b], and TensoRF [Che+22] proposed models where spatially-localized features are used as coefficients of the spherical harmonic (SH) basis, to describe view-dependent color. Such SH decoders can give both high-fidelity reconstructions and enhanced interpretability compared to MLP decoders. However, SH coefficients are difficult to optimize, and their expressivity is limited by the number of SH basis functions used (often limited 2nd degree harmonics, which produce blurry specular reflections). Instead, we replace the SH functions with a learned basis, retaining the interpretability of treating features as coefficients for a linear decoder yet increasing the expressivity of the basis and allowing it to adapt to each scene, as was proposed in NeX [Wiz+21]. We represent the basis using a small MLP that maps each view direction $\boldsymbol{d}$ to red $b_R(\boldsymbol{d}) \in \mathbb{R}^M$, green $b_G(\boldsymbol{d}) \in \mathbb{R}^M$, and blue $b_B(\boldsymbol{d}) \in \mathbb{R}^M$ *basis vectors*. The MLP serves as an adaptive drop-in replacement for the spherical harmonic basis functions repeated over the three color channels. We obtain the color values

$$\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{d}) = \bigcup_{i \in \{R,G,B\}} f(\boldsymbol{q}) \cdot b_i(\boldsymbol{d}), \tag{5.6}$$

where $\cdot$ denotes the dot product and $\cup$ denotes concatenation. Similarly, we use a learned basis $b_\sigma \in \mathbb{R}^M$, independent of the view direction, as a linear decoder for density:

$$\sigma(\boldsymbol{q}) = f(\boldsymbol{q}) \cdot b_\sigma. \tag{5.7}$$

Predicted color and density values are finally forced to be in their valid range by applying the sigmoid to $\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{d})$, and the exponential (with truncated gradient) to $\sigma(\boldsymbol{q})$.

MLP decoder: a hybrid model. Our model can also be used with an MLP decoder like that of Instant-NGP [Mül+22] and DVGO [SSC22], turning it into a hybrid model. In this version, features are decoded by two small MLPs, one g_σ that maps the spatially-localized features into density σ and additional features $\hat{f}$, and another g_{RGB} that maps $\hat{f}$ and the embedded view direction $\gamma(\boldsymbol{d})$ into RGB color

$$\begin{aligned} \sigma(\boldsymbol{q}), \hat{f}(\boldsymbol{q}) &= g_\sigma(f(\boldsymbol{q})) \\ \boldsymbol{c}(\boldsymbol{q}, \boldsymbol{d}) &= g_{RGB}(\hat{f}(\boldsymbol{q}), \gamma(\boldsymbol{d})). \end{aligned} \tag{5.8}$$

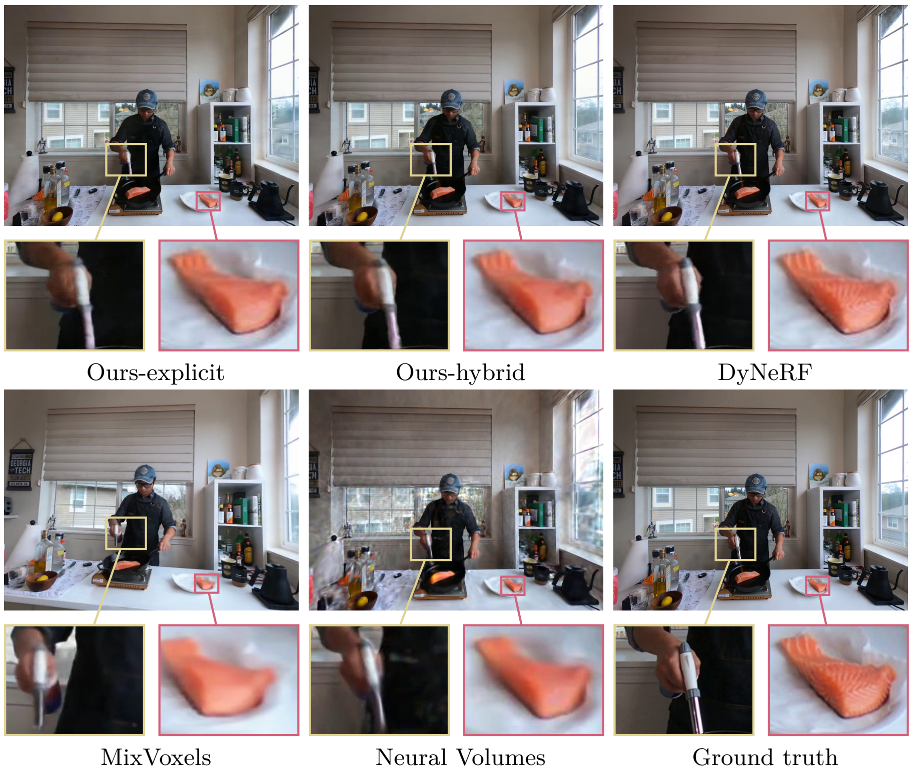


Figure 5.4: **Qualitative video results.** Our hexplane model rivals the rendering quality of state-of-the-art neural rendering methods. Our renderings were obtained after at most 4 hours of optimization on a single GPU whereas DyNeRF trained for a week on 8 GPUs. MixVoxels frame comes from a slightly different video rendering, and is thus slightly shifted.

As in the linear decoder case, the predicted density and color values are finally normalized via exponential and sigmoid, respectively.

Global appearance. We also show a simple extension of our K-planes model that enables it to represent scenes with consistent, static geometry viewed under varying lighting or appearance conditions. Such scenes appear in the Phototourism [Jin+21] dataset of famous landmarks photographed at different times of day and in different weather. To model this variable appearance, we augment K-planes with an M-dimensional vector for each training image $1, \dots, T$. Similar to NeRF-W [Mar+21], we optimize this per-image feature vector and pass it as an additional input to either the MLP learned color basis b_R, b_G, b_B, in our explicit version, or to the MLP color decoder g_{RGB}, in our hybrid version, so that it can affect color but not geometry.

5.3 Experiments

We demonstrate the broad applicability of our planar decomposition via experiments in static scenes (both bounded 360° and unbounded forward-facing), dynamic scenes (forward-facing multi-view and bounded 360° monocular), and Phototourism scenes with variable appearance. For all experiments, we report the metrics PSNR (pixel-level similarity) and SSIM[1] [Wan+04] (structural similarity), as well as approximate training time and number of parameters (in millions), in Table 5.2. Blank entries in Table 5.2 denote baseline methods for which the corresponding information is not readily available.

5.3.1 Dynamic scenes

We evaluate our hexplane model on two dynamic scene datasets: a set of synthetic, bounded, 360°, monocular videos from D-NeRF [Pum+21] and a set of real, unbounded, forward-facing, multiview videos from DyNeRF [Li+22a].

The D-NeRF dataset contains eight videos of varying duration, from 50 frames to 200 frames per video. Each timestep has a single training image from a different viewpoint; the camera "teleports" between adjacent timestamps [Gao+22]. Standardized test views are from novel camera positions at a range of timestamps throughout the video. Both our explicit and hybrid models outperform D-NeRF in both quality metrics and training time, though they do not surpass very recent hybrid methods TiNeuVox

	PSNR ↑	SSIM ↑	Train Time ↓	# Params ↓
NeRF [Mil+20] (static, synthetic)				
Ours-explicit	32.21	0.960	38 min	33M
Ours-hybrid	32.36	0.962	38 min	33M
Plenoxels [Fri+22]	31.71	0.958	11 min	∼500M
TensoRF [Che+22]	33.14	0.963	17 min	18M
I-NGP [Mül+22]	33.18	-	5 min	∼ 16M
LLFF [Mil+19] (static, real)				
Ours-explicit	26.78	0.841	33 min	19M
Ours-hybrid	26.92	0.847	33 min	19M
Plenoxels	26.29	0.839	24 min	∼500M
TensoRF	26.73	0.839	25 min	45M
D-NeRF [Pum+21] (dynamic, synthetic)				
Ours-explicit	31.05	0.97	52 min	37M
Ours-hybrid	31.61	0.97	52 min	37M
D-NeRF	29.67	0.95	48 hrs	1-3M
TiNeuVox[Fan+22]	32.67	0.97	30 min	∼12M
V4D[Gan+22]	33.72	0.98	4.9 hrs	275M
DyNeRF [Li+22a] (dynamic, real)				
Ours-explicit	30.88	0.960	3.7 hrs	51M
Ours-hybrid	31.63	0.964	1.8 hrs	27M
DyNeRF [Li+22a]	[1]29.58	-	1344 hrs	7M
LLFF [Mil+19]	[1]23.24	-	-	-
MixVoxels-L[Wan+22a]	30.80	0.960	1.3 hrs	125M
Phototourism [Jin+21] (variable appearance)				
Ours-explicit	22.25	0.859	35 min	36M
Ours-hybrid	22.92	0.877	35 min	36M
NeRF-W [Mar+21]	27.00	0.962	384 hrs	∼2M
NeRF-W (public)[2]	19.70	0.764	164 hrs	∼2M
LearnIt [Tan+21]	19.26	-	-	-

Table 5.2: **Results.** Averaged metrics over all scenes in the respective datasets. Note that Phototourism scenes use MS-SSIM (multiscale structural similarity) instead of SSIM. K-planes timings are based on a single NVIDIA A30 GPU.

[1]Note that among prior work, some evaluate using an implementation of SSIM from MipNeRF [Bar+21] whereas others use the scikit-image implementation, which tends to produce higher values. For fair comparison on each dataset we make a best effort to use the same SSIM implementation as the relevant prior work.

[Fan+22] and V4D [Gan+22],
as shown in Figure 5.5.

The DyNeRF dataset contains six 10-second videos recorded at 30 fps simultaneously by 15-20 cameras from a range of forward-facing view directions; the exact number of cameras varies per scene because a few cameras produced miscalibrated videos. A central camera is reserved for evaluation, and training uses frames from the remaining cameras. Both our methods again produce similar quality metrics to prior state-of-the-art, including recent hybrid method MixVoxels [Wan+22a], with our hybrid method achieving higher quality metrics. See Figure 5.4 for a visual comparison.

5.3.1.1 Decomposing time and space

One neat consequence of our planar decomposition of time and space is that it naturally disentangles dynamic and static portions of the scene. The static-only part of the scene can be obtained by setting the three time planes to one (the multiplicative identity). Subtracting the static-only rendered image from the full rendering (i.e. with the time plane parameters not set to 1), we can reveal the dynamic part of the scene. Figure 5.7 shows this decomposition of time and space. This natural volumetric disentanglement of a scene into static and dynamic regions may enable many applications across augmented and virtual reality [Ben+22].

We can also visualize the time planes to better understand where motion occurs in a video. Figure 5.6 shows the averaged features learned by the *xt* plane in our model for the *flame salmon* and *cut beef* DyNeRF videos, in which we can identify the motions of the hands in both space and time. The *xt* plane learns to be sparse, with most entries equal to the multiplicative identity, due to a combination of our

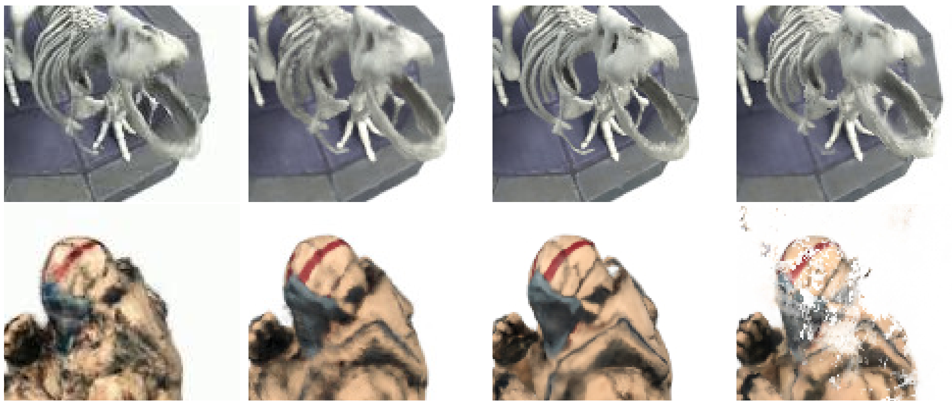

((a)) Ours-explicit ((b)) TiNeuVox ((c)) V4D ((d)) D-NeRF

Figure 5.5: **Zoomed qualitative results on scenes from D-NeRF [Pum+21].** Visual comparison of K-planes, D-NeRF [Pum+21], TiNeuVox [Fan+22] and V4D [Gan+22], on *t-rex* (top) and *hook* (bottom).

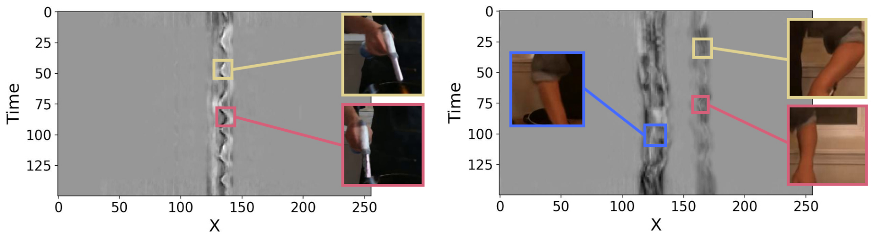


Figure 5.6: **Visualization of a time plane.** The xt plane highlights the dynamic regions in the scene. The wiggly patterns across time correspond to the motion of the person's hands and cooking tools, in the *flame salmon* scene (left) where only one hand moves and the *cut beef* scene (right) where both hands move.

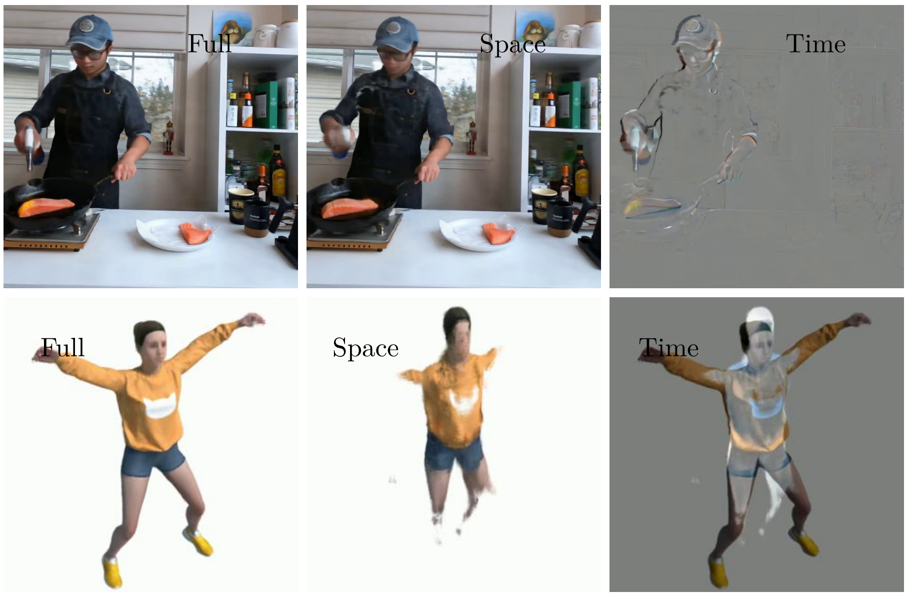


Figure 5.7: **Decomposition of space and time.** K-planes (left) naturally decomposes a 3D video into static and dynamic components. We render the static part (middle) by setting the time planes to the identity, and the remainder (right) is the dynamic part. Top shows the *flame salmon* multiview video [Li+22a] and bottom shows the *jumping jacks* monocular video [Pum+21].

sparse transients prior and the true sparsity of motion in the video. For example, in the left side of Figure 5.6 one of the cook's arms contains most of the motion, while in the right side both arms move. Having access to such an explicit representation of time allows us to add time-specific priors.

5.3.2 Variable appearance

Our variable appearance experiments use the Phototourism dataset [Jin+21], which includes photos of well-known landmarks taken by tourists with arbitrary view directions, lighting conditions, and transient occluders, mostly other tourists. Our experimental conditions parallel those of NeRF-W [Mar+21]: we train on more than a thousand tourist photographs and test on a standard set that is free of transient occluders.

Like NeRF-W, we evaluate on test images by optimizing our per-image appearance feature on the left half of the image and computing metrics on the right half. Visual comparison to prior work is shown in the appendix. Also similar to NeRF-W [Mar+21; Boj+17], we can interpolate in the appearance code space. Since only the color decoder (and not the density decoder) takes the appearance code as input, our approach is guaranteed not to change the geometry, regardless of whether we use our explicit or our hybrid model. Figure 5.8 shows that our planar decomposition with a 32-dimensional appearance code is sufficient to accurately capture global appearance changes in the scene.

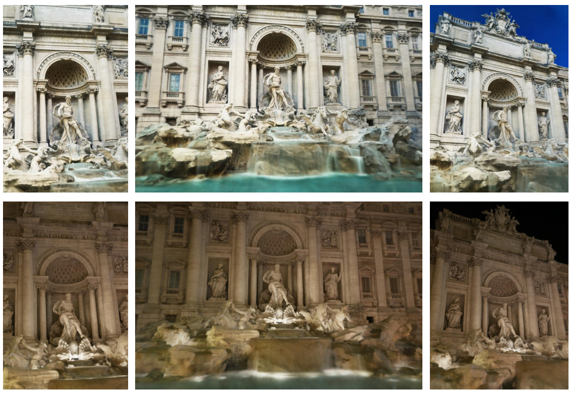


Figure 5.8: **Appearance interpolation**. Like NeRF-W [Mar+21], we can interpolate our appearance code to alter the visual appearance of landmarks. We show three test views from the *Trevi fountain* with appearance codes corresponding to day and night.

5.4 Generative modelling of 3D shapes

These NeRF models presented in the dynamic setting required extensive captures, and the rendering quality greatly deteriorate when fewer images are available during the reconstruction process or when a view far away from the training views are rendered.

Thus, casual captures of NeRFs [Mil+20] are usually of lower quality than most captures shown in NeRF papers. When a typical user (e.g., a hobbyist) captures a NeRFs, the ultimate objective is often to render a fly-through path from a considerably different set of viewpoints than the originally captured images. This large viewpoint change between training and rendering views usually reveals *floater* artifacts and bad geometry, as shown in Figure 5.9a. One way to resolve these artifacts is to teach or otherwise encourage users to more extensively capture a scene, as is commonly

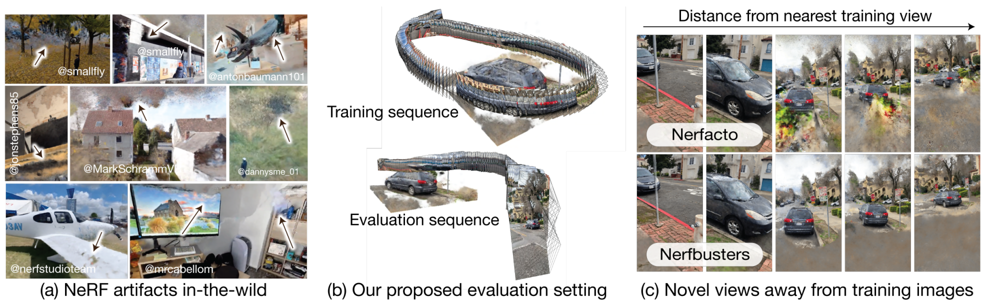


Figure 5.9: **Nerfbusters.** Rendering NeRFs at novel views far away from training views can result in artifacts, such as floaters or bad geometry. These artifacts are prevalent in in-the-wild captures (a) but are rarely seen in NeRF benchmarks, because evaluation views are often selected from the same camera path as the training views. We propose a new dataset of in-the-wild captures and a more realistic evaluation procedure (b), where each scene is captured by two paths: one for training and one for evaluation. We also propose Nerfbusters, a 3D diffusion-based method that improves scene geometry and reduces floaters (c), significantly improving upon existing regularizers in this more realistic evaluation setting.

done in apps such as Polycam[2] and Luma[3], which will direct users to make three circles at three different elevations looking inward at the object of interest. However, these capture processes can be tedious, and furthermore, users may not always follow complex capture instructions well enough to get an artifact-free capture.

Another way to clean NeRF artifacts is to develop algorithms that allow for better out-of-distribution NeRF renderings. Prior work has explored ways of mitigating artifacts by using camera pose optimization [Wan+21; Lin+21] to handle noisy camera poses, per-image appearance embeddings to handle changes in exposure [Mar+21], or robust loss functions to handle transient occluders [Sab+23]. However, while these techniques and others show improvements on standard benchmarks, most benchmarks focus on evaluating image quality at held-out frames from the training sequence, which is not usually representative of visual quality at novel viewpoints. Figure 5.9c shows how the Nerfacto method starts to degrade as the novel-view becomes more extreme.

We propose both (1) a novel method for cleaning up casually captured NeRFs and (2) a new evaluation procedure for measuring the quality of a NeRF that better reflects rendered image quality at novel viewpoints. Our proposed evaluation protocol is to capture two videos: one for training a NeRF, and a second for novel-view evaluation (Figure 5.9b). Using the images from the second capture as ground-truth (as well as depth and normals extracted from a reconstruction on *all* frames), we can compute a set of metrics on visible regions where we expect the scene to have been reasonably captured in the training sequence. Following this evaluation protocol, we

[2]`https://poly.cam/`
[3]`https://lumalabs.ai/`

capture a new dataset with 12 scenes, each with two camera sequences for training and evaluation.

We also propose *Nerfbusters*, a method aimed at improving geometry for everyday NeRF captures by improving surface coherence, cleaning up floaters, and removing cloudy artifacts. Our method learns a local 3D geometric prior with a diffusion network trained on synthetic 3D data and uses this prior to encourage plausible geometry during NeRF optimization. Compared to global 3D priors, local geometry is simpler, category-agnostic, and more repeatable, making it suitable for arbitrary scenes and smaller-scale networks (a 28 Mb U-Net effectively models the distribution of all plausible surface patches). Given this data-driven, local 3D prior, we use a novel unconditional Density Score Distillation Sampling (DSDS) loss to regularize the NeRF. We find that this technique removes floaters and makes the scene geometry crisper. To the best of our knowledge, we are the first to demonstrate that a learned local 3D prior can improve NeRFs. Empirically, we demonstrate that Nerfbusters achieves state-of-the-art performance for casual captures compared to other geometry regularizers.

Evaluating NeRFs in-the-wild. Early works in neural rendering [Mil+19], including NeRF [Mil+20], established an evaluation protocol for novel view synthesis, where every 8th frame from a camera trajectory is used for evaluation. Most follow-up works have adapted this protocol and demonstrated impressive results on forward-facing scenes in LLFF [Mil+19], synthetic scenes [Mil+20], or 360 scenes [Bar+22b; Rei+21]. In these datasets, the training and evaluation views share camera trajectories, thus the methods are evaluated only for small viewpoint changes, as illustrated in Figure 5.10. In contrast, we propose to record two camera trajectories,

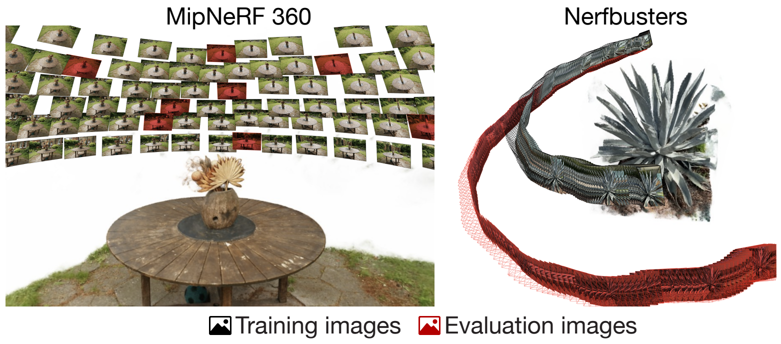


Figure 5.10: **Evaluation protocols.** Current evaluation of NeRFs (e.g., MipNeRF 360) measures render quality at every 8th frame of the captured (training) trajectory, thus only testing the model's ability to render views with small viewpoint changes. In contrast, we propose a new evaluation protocol, where two sequences are captured of the same scene: one for training and one for evaluation. Please see the supplementary material for plots showing the training and evaluation sequences for various NeRF datasets, including our proposed Nerfbuster Dataset.

one for training and one for evaluation.

We find that viewpoint changes are very limited, and the proposed Nerfbuster dataset is much more challenging. Recently, Gao et al. [Gao+22] revisited the evaluation process for dynamic NeRFs, also highlighting shortcomings in dynamic NeRF evaluation. NeRFs for extreme viewpoint changes and few-shot reconstruction have been explored on ShapeNet [Cha+15], DTU [Jen+14], and CO3D [Rei+21], where a few or just a single view is available during training. These works focus on the generalization and hallucination of unseen regions, and either assume a category-specific prior [ZT22; Yu+21a] or focus on simple scenes [Yu+21a]. In contrast, our casual captures setting assumes that a $10 - 20$ second video is available at training time, better reflecting how people capture NeRFs. We then evaluate fly-throughs with extreme novel views on an entirely different video sequence, as illustrated in Figure 5.10.

5.5 Evaluation Procedure

We propose an evaluation protocol that captures two videos, one for training and one for evaluating a NeRF. Training videos should be around $10 - 20$ seconds which are indicative of what a user might do when prompted to scan an object or scene. Anything longer than this may reduce the appeal and practicality of using NeRFs. The second video represents the novel view that a user may wish to render. The second video is only used as ground truth and does not change how users currently interact with NeRFs. We record 12 scenes (two videos each) in this way to construct our Nerfbusters Dataset. All videos were taken with a hand-held phone to approximate the casual capture setup.

Evaluating on casual captures. The steps to create our evaluation data can be boiled down to the following straightforward steps:

1. Record a video to train a NeRF (training split)
2. Record a second video with a viewpoint change (evaluation split)
3. Extract images from both videos and run SfM on all images
4. Train a "pseudo ground truth" model on both splits and save depth, normal, and visibility maps for the evaluation split.
5. Train your proposed method on the training split and evaluate with the evaluation split images and their pseudo ground truth maps.

In Figure 5.11, we show an evaluation image and its visibility, depth, and normal maps. The pseudo ground truth is high quality since it has been trained together with the first video. The visibility map is computed by taking the depth map, back-projecting each pixel into a 3D point, and then counting how many times that 3D point is seen from a training viewpoint. This dataset with associated visibility masks and processing code can be found at `https://ethanweber.me/nerfbusters`.

Masking valid regions. Rendering extreme novel views exposes part of the scene that was not captured in the training views. As most existing NeRFs are not

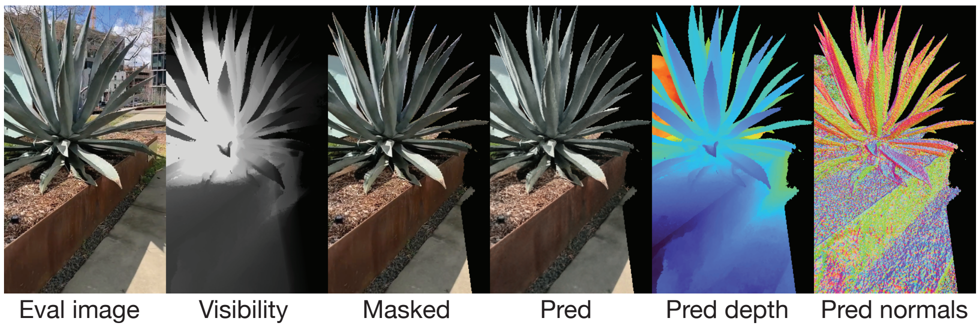


Figure 5.11: **Evaluation capture.** Here we show the data used in our evaluation protocol. The evaluation trajectory is a separate capture that is held out during the optimization of the NeRF. Individual components shown here are further described in Section 5.5.

designed to hallucinate completely unseen views, we only evaluate regions observed in the train capture trajectories using visibility masks.

More specifically, we mask out regions that are either (1) not seen by any training views (i.e., where the visibility map is zero) or (2) are predicted to be too far away (i.e., predicted depth $>$ distance threshold). We set this threshold to two times the largest distance between any two camera origins in both the training and evaluation splits. In the Nerfstudio codebase, this corresponds to a value of 2 because camera poses are scaled to fit within a box with bounds (-1,-1,-1) and (1,1,1).

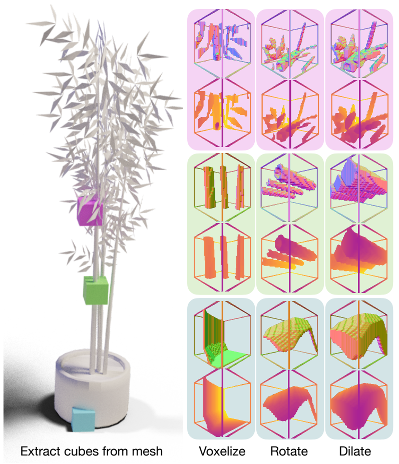


Figure 5.12: **Training data for Nerfbusters diffusion model.** Given a mesh, we extract local cubes scaled 1-10% of the mesh size. We voxelize these cubes with resolution 32^3, and augment them with random rotations and random dilation.

Coverage. Because we mask out pixels by both visibility [Gao+22] and depth, we report “coverage” which is the percent of evaluated pixels within the visible regions, commonly reported in depth completion [ZF18; War+22; WRL22]. For example, removing all densities and predicting infinite depth would result in zero coverage.

Image quality and geometry metrics. We use masked versions of PSNR, SSIM, and LPIPS for image quality. We also report on depth (MSE and mean abs. disparity difference) and normals (mean and median degrees, and the percent of valid pixels with normals < 30 degrees). We report averages for all images in the Nerfbusters Dataset in Sec. 5.7.

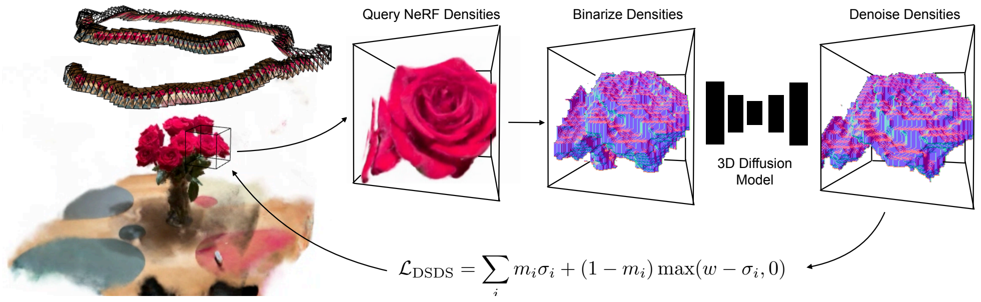

Query NeRF Densities
Binarize Densities
Denoise Densities
3D Diffusion Model
$\mathcal{L}_{\text{DSDS}} = \sum_i m_i \sigma_i + (1 - m_i) \max(w - \sigma_i, 0)$

Figure 5.13: **Method overview.** We learn a local 3D prior with a diffusion model that regularizes the 3D geometry of NeRFs. We use importance sampling to query a 32^3 cube of NeRF densities. We binarize these densities and perform one single denoising step using a pre-trained 3D diffusion model. With these denoised densities, we compute a density score distillation sampling (DSDS) that penalizes NeRF densities where the diffusion model predicts empty voxels and pushes the NeRF densities above the target w where the diffusion model predicts occupied voxels $m = \mathbb{1}\{x_0 < 0\}$.

5.6 Nerfbusters model

Our method consists of two steps. First, we train a diffusion model to denoise local 3D cubes. This model is trained on synthetic data and learns a prior over local 3D shapes. Second, we apply this local prior to real 3D scenes represented by NeRFs. We do this by querying densities in local cubes in the scene and using a novel Density Score Distillation Score (DSDS) loss to regularize our implicit scene representation. This prior improves reconstructions in regions with sparse supervision signals and removes floaters. Figure 5.13 provides an overview of our pipeline.

5.6.1 Data-driven 3D prior

Following the recent process in the context of denoising generative diffusion models [Soh+15; SE19; ND21; Rom+21; Poo+22], we formulate our generative model as a denoising diffusion probabilistic model (DDPM) [HJA20], which iteratively denoises a voxelized $32 \times 32 \times 32$ cube x of occupancy. Our diffusion model ϵ_θ is trained with

$$\mathcal{L}_{\text{Diff}} = \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|_2^2, \tag{5.9}$$

where $t \sim \mathcal{U}(0, 1000)$, $\epsilon \sim \mathcal{N}(0, I)$ and $\bar{\alpha}$ follows a linear schedule that determines the amount of noise added at timestep t. We implement our diffusion model as a small 3D U-Net [RFB15] with three downsampling layers that double the number of channels per downsampling. We train the model on synthetic 3D cubes from ShapeNet and find that a small U-Net with only 7.2 M parameters (28MB) is sufficient to learn a local 3D prior over shapes.

5.6.2 Curate synthetic 3D cubes

We train our diffusion model on local cubes sampled from ShapeNet [Cha+15], illustrated in Figure 5.12. We sample a random ShapeNet mesh and extract N local meshes at the boundary with sizes between 1-10% of the mesh min and max vertices. We voxelize these local meshes into cubes with a resolution of 32^3. We then augment the cubes with random rotations and dilation. This data processing pipeline is fast and performed online during training to increase the diversity of 3D cubes. We find that adjusting the thickness of the surface with dilation rather than infilling the mesh is faster and better defined for non-watertight meshes. Figure 5.12 illustrates the large diversity in the local cubes—some contain flat surfaces (bottom of the vase), round shapes (stem), and fine structures (leaves).

5.6.3 Applying 3D prior in-the-wild

We represent a 3D scene with a Neural Radiance Field (NeRF), [Mil+20] which takes a 3D point as input and outputs color and density, and is trained with differentiable volume rendering [Mil+20; Max95]. We build on the Nerfacto model from Nerfstudio [Tan+23] that combines recent progress in NeRFs including hash grid encoding [Mül+22], proposal sampling [Bar+21], per-image-appearance optimization [Mar+21], and scene contraction [Bar+21]. Although Nerfacto has been optimized for in-the-wild image captures, it still reveals floaters when rendered from novel views. To address these issues, we leverage the pretrained Nerfbusters diffusion model. We propose a novel sampling strategy that samples cubes from non-empty regions and a Density Score Distillation Sampling (DSDS) loss that distills the diffusion prior into the NeRF. As a result, our approach yields better scene geometry.

Importance sampling cubes. Since the NeRF represents a density field, we can query voxelized cubes in 3D space at any size, location, and resolution. For an efficient sampling of the location of the 3D cubes, we propose to store a low-resolution occupancy grid of either *accumulation weights* or *densities*. We sample the location of the 3D cubes from the distribution of this low-resolution occupancy grid. Storing *accumulation weights* in the occupancy grid yields cubes sampled mostly on frequently seen surfaces. Whereas, storing *densities* in the occupancy grid enables sampling of occluded regions. In practice, we clamp densities to one, to avoid a few densities dominating the sampling probability. We apply an exponential moving average (EMA) decay on the grid to update the occupancy grid when floaters are deleted. This importance sampling method comes with almost no added cost since we store the densities or weights along the rays already used for volume rendering, and use a small 20^3 occupancy grid. Following the sampling of a cube center location, we proceed to sample cubes of resolution 32^3 and 1-10% of the scene.

Density Score Distillation Sampling (DSDS). Our diffusion model is trained on discretized synthetic data in $\{-1, 1\}$ indicating free or occupied space, respectively.

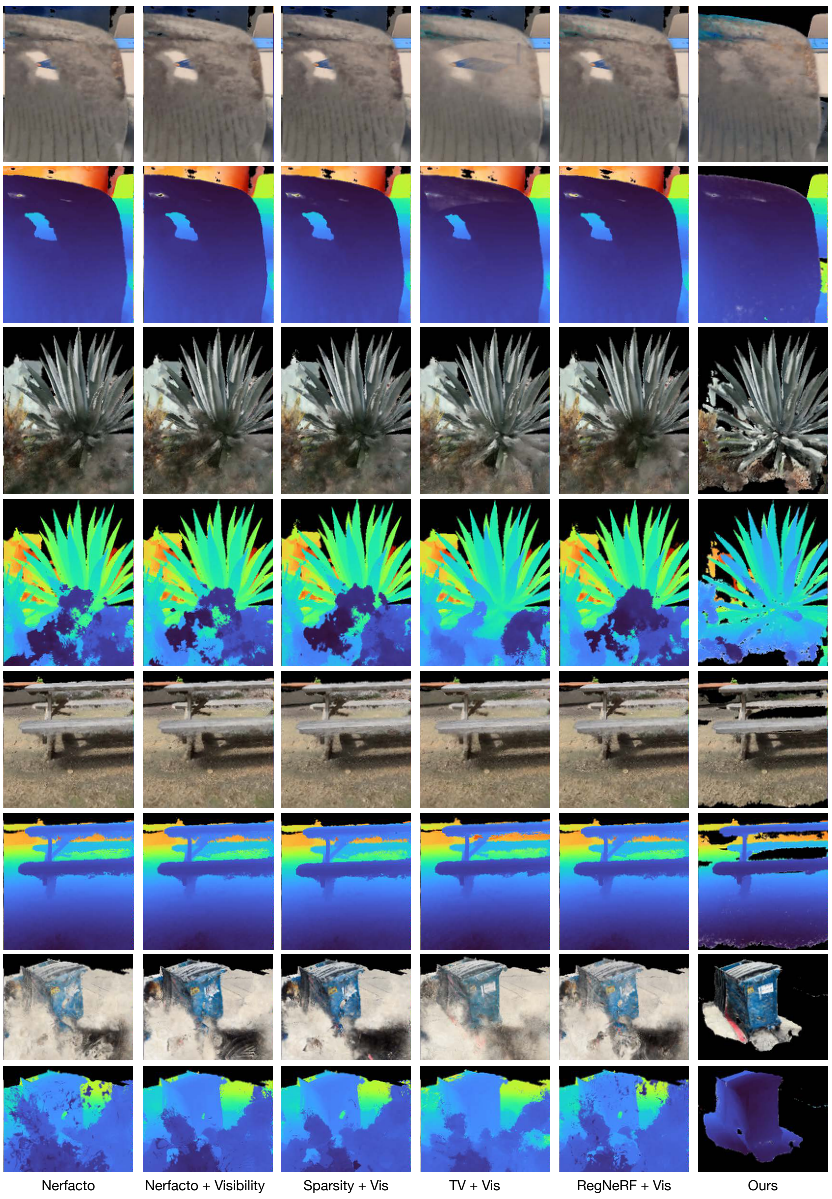


Figure 5.14: **Qualitative results.** NeRFs suffer from floaters and bad geometry when rendered away from training views. Our proposed diffusion prior fills holes (first rows), removes floaters (second and fifth row), and improves geometry (all). Please see the supplementary material for video results on our evaluation splits.

NeRF densities, on the other hand, are in $[0, \infty)$, where low densities indicate free space and larger densities mean more occupied space. In practice, we observe that densities less than 0.01 are mostly free space, whereas occupied space have density values ranging from $[0.01, 2000]$. We propose a Density Score Distillation Sampling (DSDS) loss that handles the domain gap between the densities without exploiting gradients.

Given a cube of NeRF densities σ, we discretize the densities $x_t = 1$ if $\sigma > \tau$ else -1 at time t, where τ is a hyperparameter that decides at what density to consider a voxel for empty or occupied. The Nerfbusters diffusion model then predicts the denoised cube x_0. The timestep t is a hyperparameter that determines how much noise the diffusion model should remove and can be interpreted as a learning rate. In practice, we choose a small $t \in [10, 50]$. With the denoised cube x_0, we penalize NeRF densities that the diffusion model predicts as empty or increase densities that the diffusion model predicts as occupied with

$$\mathcal{L}_{\text{DSDS}} = \sum_i m_i \sigma_i + (1 - m_i) \max(w - \sigma_i, 0), \tag{5.10}$$

where $m = \mathbb{1}\{x_0 < 0\}$ is a mask based on the denoised predictions. We penalize densities where the diffusion model predicts emptiness and increase densities where the model predicts occupancy. w is a hyperparameter that determines how much to increase the densities in occupied space. The max operator ensures that no loss is applied if an occupied voxel already has a density above w. Similar to SDS [Poo+22; Wan+22b], the DSDS loss distills the diffusion prior with a single forward pass and without backpropagating through the diffusion model.

Why not just... use a differentiable function to convert densities to the valid range of the diffusion model, then compute the SDS loss [Poo+22; Wan+22b], and then backpropagate through the activation function? This would require a function $s : \sigma \to x_t$ to map $s(0) = -1$, $s(\tau) = 0$, and $s(2\tau) = 1$, where τ is the crossing value where densities begin to be occupied. A scaled and shifted sigmoid function or a clamped linear function satisfies these requirements, but both have very steep gradients in some regions and no gradients in other regions, resulting in issues when backpropagating. In contrast, DSDS has gradients for any density predicted to be empty or occupied. In practice, we set $\tau = w = 0.01$ meaning our method deletes densities at points predicted to be empty and otherwise leaves the points unconstrained for the NeRF RGB loss to freely optimize.

Why not just... use accumulated weights, which are in the range [0, 1]? Weights are more well-behaved than densities but more expensive to compute as they require shooting a ray through the scene, evaluating and accumulating the densities along a ray. This results in significantly more function calls, but more fundamentally, requires one to specify a view from which to shoot the rays. This limits the diffusion prior to improving regions that are visible regions from the chosen view. A similar issue arises when using 2D or 2.5D priors [Nie+22; WT23], where they may not regularize occluded regions unless viewpoints are chosen in a scene-specific way.

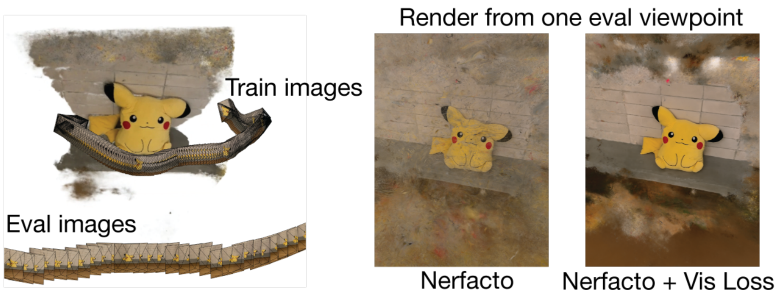


Figure 5.15: **Visibility loss.** Our visibility loss enables stepping behind or outside the training camera frustums. We accomplish this by supervising densities to be low when not seen by at least one training view. Other solutions would be to store an occupancy grid [Mül+22] or compute ray-frustum intersection tests during rendering. Our solution is easy to implement and applicable to any NeRF.

5.6.4 Visibility Loss

Our proposed local 3D diffusion model improves scene geometry and removes floaters, but it requires decent starting densities since it operates locally and thus needs contextual information to ground its denoising steps. To this end, we propose a simple loss that penalizes densities at 3D locations that are not seen by multiple training views. We find this simple regularizer effective in removing floaters from regions outside the convex hull of the training views. We define our visibility loss as

$$\mathcal{L}_{\text{vis}} = \sum_i V(q_i) f_\sigma(q_i), \tag{5.11}$$

where $f_\sigma(q_i) = \sigma_i$ is the NeRF density at the 3D location q_i, and $V(q_i) = \mathbb{1}\{\sum_{j=1} v_{ij} < 1\}$ indicates if the location is not visible from any training views. We approximate the visibility $v_{ij} \in \{0, 1\}$ of the i'th 3D location in the j'th training view with a frustum check. This approximation does not handle occlusions, instead overestimates the number of views a location is visible from. This loss penalizes densities in regions not seen by training images.

In practice, we implement this by defining a single tight sphere around our training images and render batches of rays that shoot from a random location on the sphere surface, through the center of the scene, and far off into the distance. We render rays with Nerfacto and apply this loss to the sampled points. Nerfacto uses a proposal sampler [Bar+22b] to importance sample around surfaces, so our loss is effective in quickly culling away any floating artifacts with high density outside visible regions. See Figure 5.15 for a qualitative result where we render from behind training images.

	PSNR ↑	SSIM ↑	LPIPS ↓	Depth ↓	Disp. ↓	Mean ° ↓	Median ° ↓	% 30° ↑	Coverage ↑
Nerfacto Pseudo GT	25.98	0.8591	0.1019	0.0	0.0	0.0	0.0	1.0	0.893
Nerfacto	17.00	0.5267	0.3800	126.277	1.510	60.63	54.638	0.254	**0.896**
+ Visibility Loss	17.81	0.5538	0.3432	100.057	1.041	57.73	51.335	0.280	0.854
+ Vis + Sparsity [Yu+21b]	17.81	0.5536	0.3445	92.168	1.145	57.77	51.399	0.280	0.854
+ Vis + TV [Fri+22]	17.84	0.5617	0.3409	74.015	0.382	61.93	56.164	0.242	0.843
+ Vis + RegNeRF [Nie+22]	17.49	0.5396	0.3585	182.447	1.200	59.39	53.267	0.268	0.858
+ Vis + DSDS (Ours)	**17.99**	**0.6060**	**0.2496**	**54.453**	**0.114**	**54.77**	**47.981**	**0.295**	0.630

Table 5.3: **Quantitative evaluation.** NeRFs suffer when rendered away from the training trajectories. Existing regularizers do not suffice to improve the geometry. Nerfbusters learns a local 3D prior with a diffusion model, which removes floaters and improves the scene geometry. Results are averaged across 12 scenes.

Cube sampling strategies

	PSNR	SSIM	Disp.	Mean °	Cov.
Uniform	14.61	0.4276	10.288	61.52	**0.886**
Densities σ	**16.46**	**0.5086**	**0.081**	**49.21**	0.606
Weights	15.86	0.4466	0.112	53.09	0.634

Activation functions

	PSNR	SSIM	Disp.	Mean °	Cov.
Clamp+SDS	12.53	0.2652	2.065	87.33	**1.000**
Sigmoid+SDS	12.53	0.2652	2.065	87.33	**1.000**
$\sigma_\mathcal{T}$+DSDS	**15.86**	**0.4466**	**0.112**	**53.09**	0.634

Cube size range as % of scene

	PSNR	SSIM	Disp.	Mean °	Cov.
1-20%	**17.05**	**0.5005**	**0.083**	54.87	0.600
10-20%	16.93	0.4884	0.090	**50.78**	**0.640**
1-10%	15.86	0.4466	0.112	53.09	0.634

Table 5.4: **Ablation study.** Ablation on the “garbage” scene for different settings of using our 3D prior as a NeRF loss. Cube sampling refers to uniformly sampling the entire scene versus importance sampling with accumulated weights or densities.

5.7 Experiments

We follow our proposed protocol described in detail in Sec. 5.5. We apply different regularizers as a post-processing approach to clean up NeRFs and also run ablations on our proposed method.

Implementation details. For each experiment, we use the Nerfacto model within the Nerfstudio [Tan+23] codebase. We turn off pose estimation for evaluation purposes and then train Nerfacto for 30K iterations which takes up to half an hour. We then fine-tune from this checkpoint with different regularizer methods. We compare the proposed method with vanilla Nerfacto, Nerfacto with the proposed visibility loss, Nerfacto with our visibility loss and 3D sparsity loss [Yu+21b], 3D TV regularization

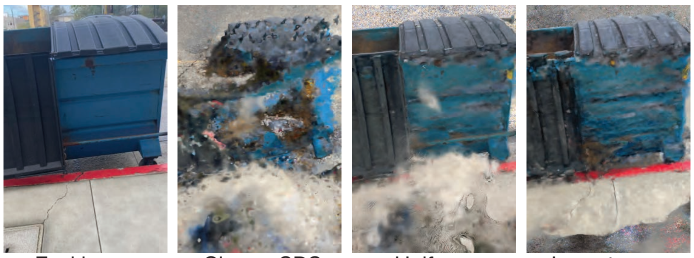


Figure 5.16: **Ablations results.** Using a simple activation function and SDS results in a not-well-behaved gradient signal, increasing the number of floaters in the scene. Importance sampling more effectively applies the 3D cube loss in space, cleaning up floaters and improving the scene geometry.

[Fri+22], and 2D TV which is RegNeRF [Nie+22]. Our implementations also use the distortion loss [Bar+22b] which is on by default with Nerfacto. All methods are effective within the first 1K iterations of fine-tuning (∼4 minutes on an NVIDIA RTX A5000 for Nerfbusters), but we train for 5K iterations. For the 3D baselines, we sample 40 32^3 cubes per iteration and for the 2D baseline RegNeRF, we render ten 32^2 patches. The usual NeRF reconstruction loss is also applied during fine-tuning with 4096 rays per batch.

Results. Table 5.3 shows that visibility loss improves vanilla Nerfacto across all quality metrics. Existing hand-crafted regularizers do not improve upon this baseline. In contrast, our learn local diffusion prior removes floaters and improves the scene geometry, yielding state-of-the-art results on these challenging casual captures. The proposed method deletes floaters, and thus we find that it has lower coverage than the baselines. Figure 5.14 shows a qualitative comparison of the methods for both indoor and outdoor scenes. We find that our method improves geometry by completing holes (see the chair in the first row), removing floaters (see in front of century plant in the second row and garbage truck in the fourth row), and sharping geometry (see the under the bench in the third row).

Ablations of our 3D prior on real data We ablate our method on the "garbage" scene (Table 5.4). We find that the cube sampling strategies (i.e., where to apply the diffusion prior) are important, and using the proposed importance sampling with densities yields the best performance. Figure 5.16 compares uniform sampling with importance sampling (using densities). Importance sampling samples less empty space, and thus is more effective at cleaning up floaters and scene geometry. We compare the proposed DSDS loss against SDS with either a scaled and shifted sigmoid or a clamped sigmoid that satisfies our requirements (see Section 5.6.3). We find the gradients do not flow well through this activation function resulting in a distorted scene

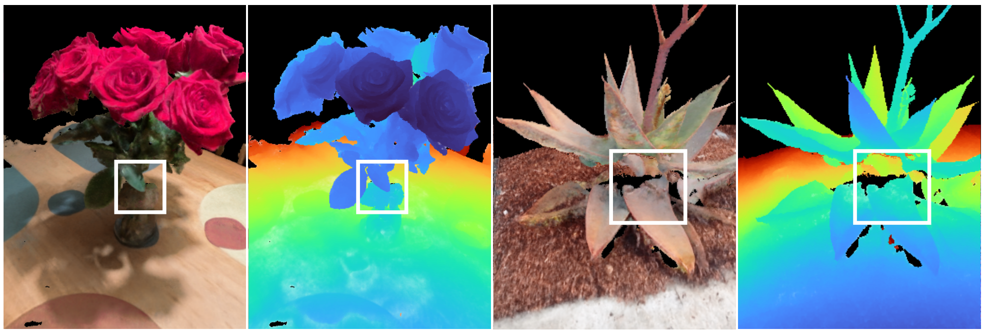


Figure 5.17: **Limitations.** The proposed model only operates on densities, which comes with some limitations. We find that it cannot distinguish floaters from transparent objects (left). It does not hallucinate texture and thus ends up removing regions that are occluded in all training views (right).

with many floaters (see Figure 5.16 left). We also ablate the cube sizes used cubes size ranging from 1% to 20% of the scene scale. We find that our method is relatively robust to the cube sizes, yielding a trade-off between removing more with larger cubes and removing less with smaller cubes.

Transparent objects. NeRFs are able to represent transparent objects by assigning low densities to the transparent object. These transparent densities behave similarly to floaters, and it requires semantic information to distinguish the two. Since our local diffusion prior does not have semantic information, it removes transparent objects as illustrated in the vase in Figure 5.17.

Hallucinating texture. The proposed method cleans geometry but cannot edit texture, as our method operates on densities. This means that we can remove regions that contain floaters or fill holes, but we cannot colorize these regions. We leave colorization and inpainting low-confidence regions to future work, where 2D diffusion priors [Poo+22; Mel+23] or 3D-consistent inpainting [Liu+21; Li+22b] may be relevant.

5.8 Summary

In this chapter, we explored NeRFs to reconstruct 3D models containing dynamic scenes or models casually captured. The main conclusions summarize as follows:

Dynamic We introduced a simple yet versatile method to decompose a d-dimensional space into $\binom{d}{2}$ planes, which can be optimized directly from indirect measurements and scales gracefully in model size and optimization time with increasing dimension.

NeRF evaluation. We propose a new evaluation procedure of Neural Radiance Fields (NeRFs) that better encompasses how artists or hobbyists use the technology.

Generative priors. We present a data-driven, local 3D diffusion prior, Nerfbusters, that removes floaters and improves the scene geometry.

CHAPTER 6

3D Reasoning

The previous chapters addressed the creation of a dense 3D map from a collection of images. Once we have such a dense 3D model, we want to be able to interact and manipulate it. The ability to interact with a 3D environment is of fundamental importance for many augmented reality (AR) application domains such as interactive visualization, entertainment, games, and robotics [ML14]. Such interactions are often semantic in nature, capturing specified entities in a 3D scene and manipulating them accordingly. In this chapter, we first present a novel framework for (1) semantically disentangling parts of a scene and then (2) propose several objectives for manipulating the disentangled parts.

This chapter contains parts from Volumetric Disentanglement for 3D Scene Manipulation [Ben+22]. Not all experiments, results, and related works are presented to keep the text shorter.

6.1 Disentangled Object Representation

We first propose a method to disentangle a NeRF representation into a foreground and background scene in a semantically consistent manner. We propose to train two NeRFs [Mil+20], one to represent the full scene and another to represent the background. We subtract these two representations to obtain the foreground object as illustrated in Figure 6.1. Then, we can manipulate this disentangled foreground object and compose it back into the 3D representation, ensuring semantically consistent volumetric scene edits.

The full volume can be trained with the standard training proposed in [Mil+20]. We train the background volume by minimizing a masked reconstruction loss between real and rendered ray color

$$\mathcal{L}_{bg}(r) = ||(1 - m(r)) \odot (c(r) - \hat{c}(r))||_2^2, \tag{6.1}$$

where $c(r)$ is the photographed color of a ray, $\hat{c}(r)$ is the rendered color of a ray, and $m(r)$ indicates if the ray belongs to the foreground object. With this training loss, we can learn a background NeRF only using 2D masks as an additional input.

The foreground object can then be found using the principle of volume mixing [DCH88]:

$$c_{fg}(r) = \sum_{i=1}^{N} w_{fg}^i \cdot c_{fg}^i \qquad w_{fg}^i = w_{full}^i - w_{bg}^i \qquad c_{fg}^i = c_{full}^i - c_{bg}^i \tag{6.2}$$

where $c_{fg}(r)$ is the foreground volume color at the pixel location corresponding to ray r, w_{bg}^i and c_{bg}^i are the estimated weights and colors of the background volume and w_{full}^i and c_{full}^i are the weights and colors of the full volume.

Object Controls. The camera parameters, poses, rays, and sampled points along the rays are chosen to be identical for both the full volume and the background volume, and hence also identical to the foreground volume. Given this canonical setting, the corresponding points along the rays for both the foreground and background can be easily found. This natural correspondence allows us to modify either the foreground volume or the background volume and then recombine into a modified rendering

$$c'(r) = \sum_{i=1}^{N} \bar{w}_{bg}^i \cdot \bar{c}_{bg}^i + \bar{w}_{fg}^i \cdot \bar{c}_{fg}^i, \tag{6.3}$$

where $c'(r)$ is the recombined color, $\bar{w}_{fg}^i$, $\bar{c}_{fg}^i$, $\bar{w}_{bg}^i$, $\bar{c}_{bg}^i$ are the modified weights and colors for the foreground and background volumes. In our experiments, we only modify the foreground and so $\bar{w}_{bg}^i = w_{bg}^i$, $\bar{c}_{bg}^i = c_{bg}^i$.

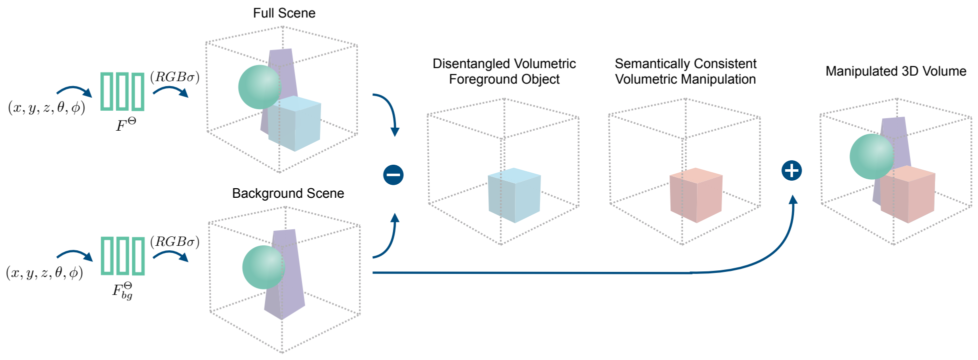

Full Scene
(x, y, z, θ, ϕ)
F^{Θ}
$(RGB\sigma)$
Background Scene
(x, y, z, θ, ϕ)
F^{Θ}_{bg}
$(RGB\sigma)$
Disentangled Volumetric Foreground Object
Semantically Consistent Volumetric Manipulation
Manipulated 3D Volume

Figure 6.1: **Overview of our disentanglement framework.** We learn a volumetric representation of the background and full scene using NeRF. We subtract the full and the background volumes to obtain a disentangled foreground volume. Then, we can manipulate the foreground volume without changing the background. We can place the foreground object back into the original scene by adding it volumetrically to the background scene, obtaining a manipulated scene.

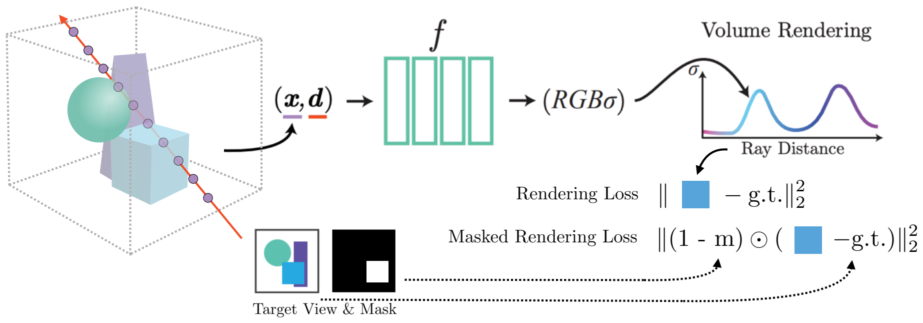

$(\boldsymbol{x}, \boldsymbol{d})$
f
$(RGB\sigma)$
Volume Rendering
σ
Ray Distance
Rendering Loss $\|\ 🟦\ - \text{g.t.}\|_2^2$
Masked Rendering Loss $\|(1 - \text{m}) \odot (\ 🟦\ -\text{g.t.})\|_2^2$
Target View & Mask

Figure 6.2: **Training losses for the background and full scene.** We train a neural radiance field on the *full scene*. To train the *background scene*, we apply a masked rendering loss, where regions that are projected inside a 2D mask $(1 - m)$, are not penalized in the loss. The network learns to reconstruct this region based on correlated effects, such as how light from the surrounding affects the masked regions, and multi-view geometry, where the background might not be masked from another view.

6.2 Object Manipulation

Given the ability to control the foreground and background volumes separately, we now propose a set of downstream manipulation tasks that emerge from our disentangled representation. As noted in Section 6.1, we can now control the weights, colors as well as translation parameters separately for the foreground and background volumes and so introduce a set of manipulation tasks that use the controls. The task of *Object Removal* is equivalent to displaying the background volume in isolation.

Object Transformation. Due to the alignment of camera parameters, as well as chosen poses, rays, and sampled points along the rays, one can apply a transformation on the background and foreground volumes separately, before recombining the volumes together. For either the foreground or the background, and for a given transformation T (for example, a rotation or translation), we simply evaluate the color and weight of point p using f_θ at position $T^{-1}(p)$ and then recombine the volumes together using (6.3).

Object Camouflage. We can change the texture of the foreground 3D object such that it is difficult to detect from its background [Owe+14; Guo+22]. Such an ability can be useful in the context of diminished reality [MIS17]. We fix the depth of the foreground object while manipulating its texture. As the depth of the foreground is derived from w^i_{fg}, we fix $\bar{w}^i_{fg} = w^i_{fg}$ and only optimize $\bar{c}^i_{fg}$. We follow (6.3), in compositing the foreground and background volumes. Let the resulting output of a ray be $\hat{c}(r)$, and let $\hat{c}_{bg}(r)$ be the corresponding output for the background volume.

We optimize a neural radiance field for foreground colors $\bar{c}^i_{fg}$ with the following loss:

$$\mathcal{L}_{camouflage}(r) = ||\hat{c}(r) - \hat{c}_{bg}(r)||_2^2 \tag{6.4}$$

As the depth is fixed, only the foreground object colors are changed so as to match the background volume as closely as possible.

Non-negative 3D Inpainting. We consider the setting of non-negative image generation [Luo+21], where we perform non-negative changes to the full scene to most closely resemble the background volume. This constraint is imposed in optical-see-through devices that can only add light to an image. In this case, we learn a residual volume to render rays $\hat{c}_{residual}(r)$ with the following loss:

$$\mathcal{L}_{non-negative}(r) = ||\hat{c}_{full}(r) + \hat{c}_{residual}(r) - \hat{c}_{bg}(r)||_2^2 \tag{6.5}$$

where $\hat{c}_i^{full}$ are rendered ray of the full scene. That is, we learn a residual volume, such that when added to the full volume, it closely resembles the views of the background.

Semantic Manipulation. We consider a mechanism for the semantic manipulation of the foreground while adhering to the global semantics of the entire scene. To this end, we consider the recently proposed model of CLIP [Rad+21], a multi-modal embedding method that can be used to find the perceptual similarity between images and texts. One can use CLIP to embed an image I and text prompt t, and to subsequently compare the cosine similarity of the embeddings. Let this operation be $sim(I, t)$, where a value of 1 indicates perceptually similar of a text and image. We note that one can also use CLIP to compare the perceptual similarity of two images I_1 and I_2, denoted $sim(I_1, I_2)$.

Let $\hat{x}$ be the result of applying (6.3) to form an entire image, while fixing the background colors and weights as well as the foreground weights. That is, we only optimize the foreground colors $\bar{c}^i_{fg}$. Similarly, we let $\hat{x}_{bg}$ to be a rendered view from the background NeRF. For a user-specified target text t, we consider the following objective:

$$\begin{aligned}\mathcal{L}_{semantic} =& 1 - sim\left(\hat{x} \odot m + \hat{x}_{bg} \odot (1-m), t\right) && (6.6)\\ &+ 1 - sim\left(\hat{x} \odot m + \hat{x}_{bg} \odot (1-m), \hat{x}_{bg} \odot (1-m)\right) && (6.7)\\ &+ ||\hat{x} \odot (1-m) - \hat{x}_{bg} \odot (1-m)||_2^2 && (6.8)\end{aligned}$$

The CLIP similarity only improves by making local changes that occur within the masked region of the foreground object, but it can 'see' the background as well as the foreground for context. We enforce that the generated volume views are similar to both the target text (6.6) and the background (6.7). To further enforce that no changes are made to the background, we constrain the background of the combined volume views to match those of the background using (6.8).

6.2.1 Training and Implementation Details

For training, we consider the natural non-synthetic scenes given in [Mil+20], together with their associated pose information. An off-the-shelf segmentation or manual annotation is used to extract masks. The method is robust to inaccurate masks such as those provided by off-the-shelf networks. Our rendering resolution for training the background and full scenes is 504×378. For the manipulation tasks, the same resolution is used for *3D inpainting*, *object camouflage*, *transformation* and *non-negative inpainting* tasks. For the *semantic manipulation* task, our rendering resolution is 252×189. For the CLIP [Rad+21] input, for a given view, we sample a 128×128 grid of points from the 252×189 output, and then upsample it to 224×224, which is the required input resolution of CLIP. We normalize the images and apply a text and image embedding as in CLIP [Rad+21]. We follow NeRF [Mil+20], in optimizing both a “coarse” and “fine” networks for a neural radiance field, and follow the same sampling strategy of points along the ray. All neural fields are parametrized using an MLP with ReLU activation of the same architecture of [Mil+20]. We use an Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with a learning rate that begins with 5×10^{-4} and decays exponentially to 5×10^{-5}.

6.3 Experiments

We divide the experimental section into two parts. First, we show that we can successfully disentangle the foreground and background volumes from the rest of the scene. Second, we demonstrate some of the many manipulation tasks this disentanglement enables, as described in Section 6.2. Corresponding 3D scenes from multiple views are provided in the project webpage. As far as we can ascertain, no other framework enables all applications we consider at once, in a simple and intuitive manner.

6.3.1 Object Disentanglement

Figure 6.3 shows views from different scenes where we separate the full scene, background, and foreground in a volumetrically and semantically consistent manner. As can be seen, the disentangled objects are consistent across views. Figure 6.4 shows how the removal of a leaf, a T-rex, and a whiteboard is consistent across multiple views. The background neural radiance field is made plausible predictions of the background scene via multi-view geometry and based on the correlated effects from the scene, *e.g.* the background behind the leaf or the legs of the T-rex might be occluded by the 2D mask from one view, but visible from another. However, the background behind the whiteboard is occluded from every angle. Nevertheless, the background neural radiance field makes a plausible prediction of the background based on the correlated effects from the surrounding scene. Further, our model can handle the disentanglement of non-planar objects, such as the T-rex, well.

	Object Disentanglement			Object Manipulation		
	Ours	DeepFill-v2 [Yu+19]	EdgeConnect [Naz+19]	Ours	GLIDE [Nic+21]	Blended [ALF21]
Q1	**3.86**	2.44	2.37	**3.85**	1.10	1.26
Q2	**3.84**	1.52	1.86	**3.78**	1.20	1.26

Table 6.1: A user study performed for the tasks of Object Disentanglement and 3D Object Manipulation. A mean opinion score (1-5) is shown where users were asked: (Q1) "How well was the desired task performed?" (object removed or semantically manipulated) and (Q2) "How realistic is the resulting scene?"

As far as we can ascertain, the closest 2D task to object disentanglement is that of object inpainting. We consider two prominent baselines of DeepFill-v2 [Yu+19] and EdgeConnect [Naz+19] for this task and compare our method on the scenes of leaves and whiteboard removal as in Fig. 6.4. We train the baseline on the same training images and their associated masks. In order to compare our method on the same novel views, we train a NeRF [Mil+20] on the resulting outputs, resulting in a scene with the same novel views as ours. Unlike our method, the results have 3D inconsistencies, artifacts, and flickering between views. Since there do not exist ground truth images, standard metrics such as PNSR/SSIM are not applicable. We, therefore, conduct a user study and ask users to rate from a scale of $1-5$: (Q1) "How well was the object removed?" and (Q2) "How realistic is the resulting scene?" We consider 25 users and mean opinion scores are shown in Table 6.1.

6.3.2 Object Manipulation

Foreground Transformation. We consider the ability to scale the foreground object and place the rescaled object back into the scene by changing the focal length used to generate the rays of the foreground object, and then volumetrically adding it back into our background volume. Figure 6.5 shows an example where the disentangled TV is twice as large. We note that other transformations such as translation and rotation are possible in a similar manner. Figure 6.5 highlights several properties of our volumetric disentanglement volume. First, the network is able to "hallucinate" how a plausible background looks in regions occluded across all views, *e.g.* behind the TV. It does this based on correlated effects from the rest of the scene. A second property is that it can disentanglement correlated effects such as the reflections on the TV screen, which is evident from the almost completely black TV in the foreground scene. Lastly, these correlated effects result in consistent and photo-realistic reflections, when we place the rescaled TV back into the scene. These reflections are consistent across views.

Object Camouflage. Another manipulation of interest is of camouflaging an object [Owe+14; Guo+22], *i.e.* only change the texture of the object and not its shape. Figure 6.6 illustrates examples of camouflaging with fixed depth, but free texture

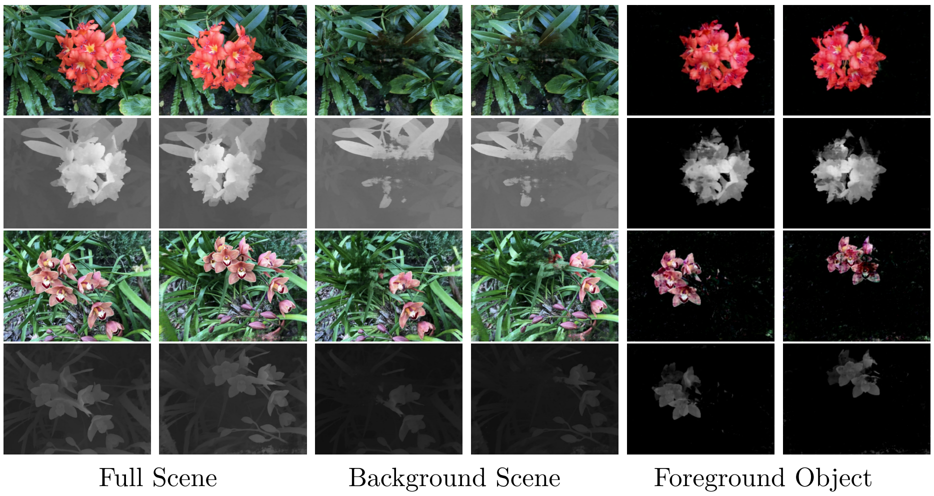


Figure 6.3: **Two rendered views of the full scene, background and foreground.** As foreground is obtained by subtracting the background from the full scene volumetrically (Section 6.1), we also obtain the disparity of the foreground.

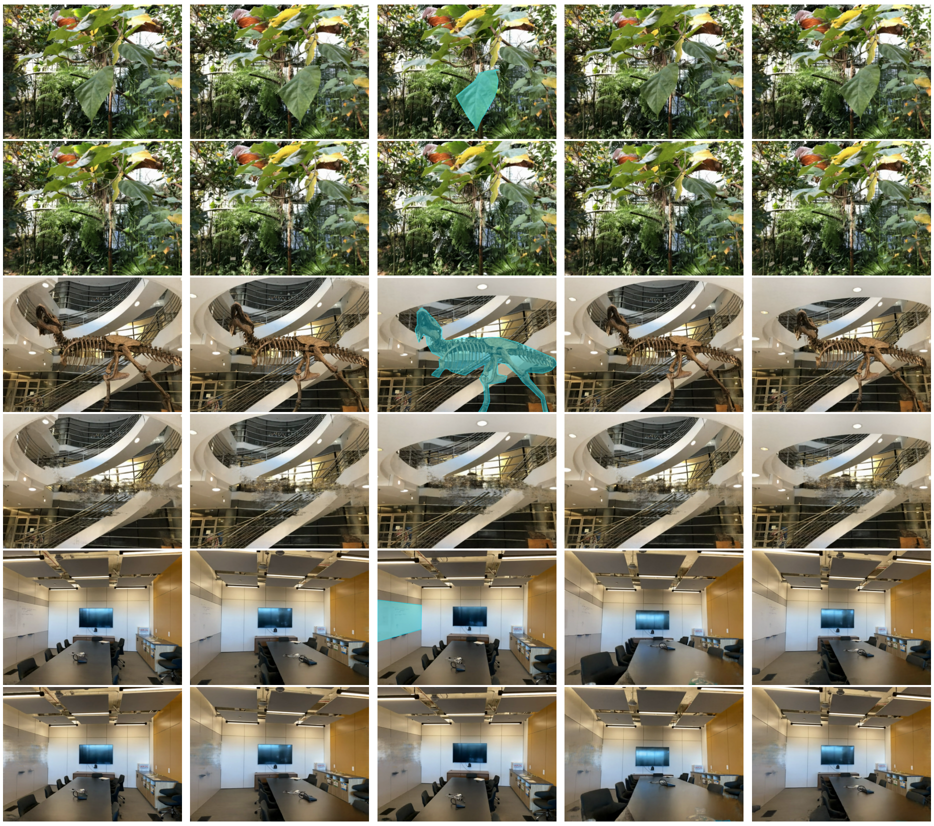


Figure 6.4: **Semantic consistency across views.** Uniformly sampled renderings of the full and the background volumes for three different scenes. The removed object is visually enhanced in the center column by a 2D mask.

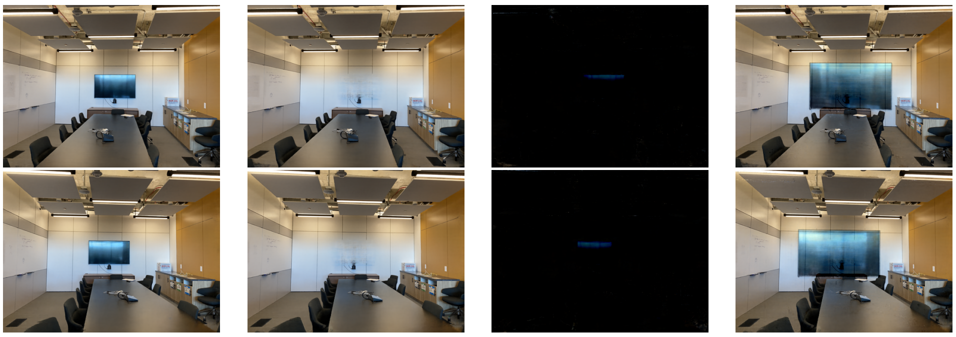

((a)) Original Scene ((b)) Background ((c)) Foreground Obj. ((d)) New Scene

Figure 6.5: **Foreground object transformation**. Our method makes plausible predictions in occluded regions (behind the TV) by understanding the correlated effects from the rest of the scene, such as the light reflections in the TV screen, which are not visible in the foreground. After scaling the foreground object and placing it back into the scene, the correlated effects are introduced again, resulting in photo-realistic and view consistent light reflections on the TV screen.

changes. While the depth of the camouflaged object and that of the foreground object match, the appearance of the camouflaged object is that of the background.

Non-Negative Inpainting. In optical see-through AR, one might also wish to camouflage objects [Luo+21] or inpaint them. However, in see-through AR one can only add light. Figure 6.7 shows how adding light can make the appearance of camouflage in a 3D consistent manner.

3D Object Manipulation. We now demonstrate how our disentanglement can be used for 3D object manipulation. Figure 6.8 shows two views of a fern. We have disentangled both the window mullion in the upper left corner and the tree trunk from the rest of the scene. Even though the window mullion is occluded in the first view, and thus our 2D mask is masking the occluding leaf in front of the window mullion, this occluding object is not part of the disentangled window mullion object. The 3D manipulations are shown in (c)-(e) in Figure 6.8. The manipulated 3D objects are semantically consistent across views. For the strawberry manipulation in (e), we see that part of the tree trunk has been camouflaged to more closely resemble the shape of a strawberry. We compare to 2D text-based inpainting methods of GLIDE [Nic+21] and Blended Diffusion [ALF21], where we follow the same procedure as in Section 6.3.1. We consider a similar user as detailed in Section 6.3.1, where Q1 is modified to: "How well was the object semantically manipulated according to the target text prompt?" For the user study we consider the fern scene of Figure 6.8, for the text prompts of "strawberry" and "old tree".

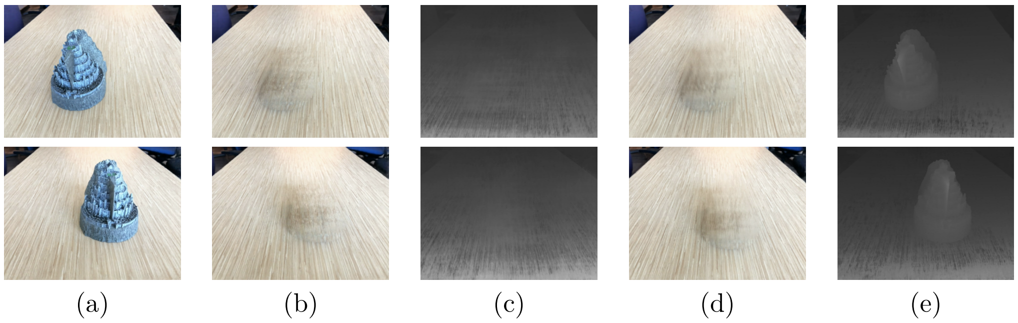


Figure 6.6: **Object camouflage for two different random views of a fortress scene.** (a) original scene, (b) background scene, (c) disparity map of the background scene, (d) camouflaged scene, (e) disparity map of camouflaged scene.

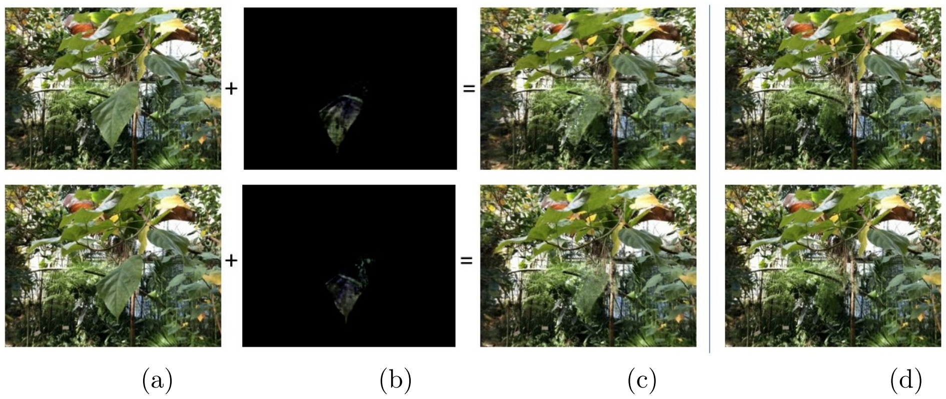


Figure 6.7: **Non-negative object inpainting for two views for a scene of leaves.** Given the full scene (a), a residual scene is added (b) resulting in scene (c), with the aim of being close to the background without the leaf (d).

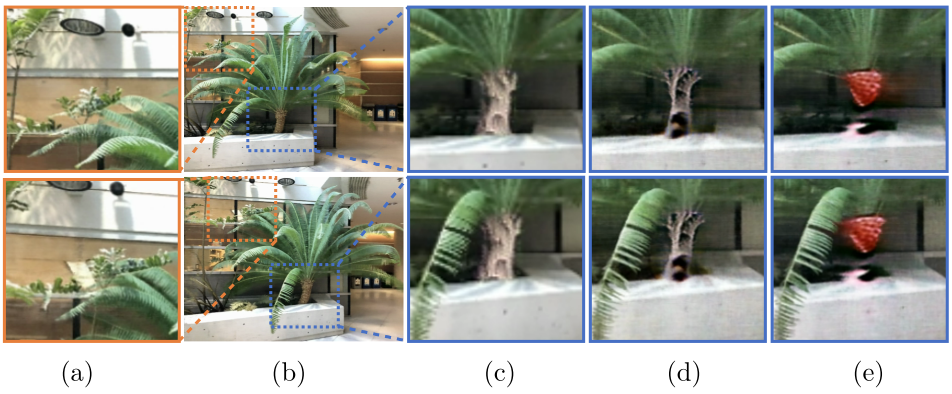


Figure 6.8: **3D Object Manipulation.** Insets of the disentangled (a) window mullion and manipulated (c)-(e) tree trunk in the original scene (b). Note how the window mullion is removed without removing the leaf of the fern that occludes it from the first view. The query text to manipulate the trunks are (c) *Old tree*, (d) *Aspen tree*, and (e) *Strawberry*. The manipulated objects are view-consistent.

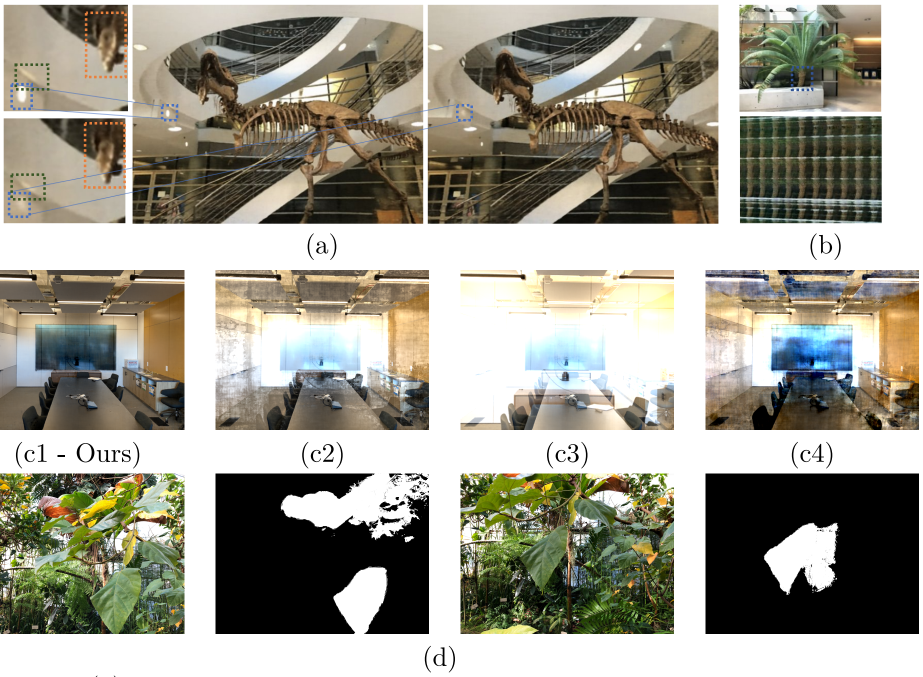


Figure 6.9: (a) *Failure to completely remove a light source.* The original light source is shown in blue in the middle image and for the background, using our method, on the right. In orange and green are regions affected by the light source, resulting in the failure to completely remove it. (b) *Illustration of the result of training a neural radiance field on the masked foreground region.* (c1-c4) *Ablation for composition.* Alternatives to the composition shown in (6.3) for foreground object translation (Figure 6.5). (d) *Robustness to noisy 2D masks.* Our method can handle noisy 2D masked obtained automatically.

6.3.3 Discussion and Limitations

Our work has some limitations. When light from the background affects the foreground object, we correctly disentangle the illuminations on the object. However, when the object is a light source, we cannot completely disentangle the object as seen in Figure 6.9(a). While our work can handle noisy masks, we require masks for all training views. We leave the task of reducing the number of masks for future work.

Another limitation is with respect to the semantic manipulation of foreground objects. We found that manipulating too large objects results in an under-constrained optimization because the signal provided by CLIP is not sufficient. For smaller objects, the background from multiple views provides a much-needed context for CLIP to provide a useful signal for the optimization. Our work is orthogonal to recent speed and generalization extensions of NeRF that could combined with our method. The number of 2D masks required by our method is also upper bounded by the number of training views and so methods such as DietNeRF [JTA21] could be combined with our method to reduce the number of 2D masks required. We leave such combinations to future work.

In Figure 6.9(b), we illustrate the necessity of extracting the foreground volume from the background and the full volume, rather than directly learning the foreground volume. We tried to train a NeRF to reconstruct the foreground volume directly, which resulted in an under-constrained optimization.

In Fig. 6.9 (c1 to c4), we show, for the task of foreground object translation (Figure 6.5), alternatives to the recombining method of (6.3), with (c2) $\bar{c}^i_{full}$ instead of $\bar{c}^i_{fg}$, (c3) $\bar{w}^i_{full}$ instead of $\bar{w}^i_{fg}$, (c4) $c(r) = \sum_{i=1}^{N}(\bar{w}^i_{bg} + \bar{w}^i_{fg}) \cdot (\bar{c}^i_{bg} + \bar{c}^i_{fg})$.

Lastly, our method can handle noisy annotations of the foreground. In Figure 6.9(d), we demonstrate the masks used for the leaves scene, which were extracted using an off-the-shelf segmentation algorithm.

6.4 Summary

In this chapter, we presented a framework for volumetric disentanglement of foreground objects from a background scene. The disentangled foreground object is obtained by volumetrically subtracting a learned volume representation of the background with one from the entire scene. The foreground-background disentanglement adheres to object occlusions and background effects such as illumination and reflections. We established that our disentanglement facilitates separate control of color, depth, and transformations for both the foreground and background objects. This enables a wide range of applications, of which we have demonstrated those of foreground transformations, object camouflage, non-negative generation, and 3D object manipulation.

CHAPTER 7

Discussion & Conclusion

This thesis explores several of the core technologies for creating a 3D model from a collection of images, namely image retrieval, structure from motion, multiview stereo, and 3D reasoning. The thesis contributes with more efficient and accurate methods while incorporating uncertainties at several steps along this pipeline. These contributions allow for more robust systems and reduce the risk of propagating errors through the reconstruction pipeline. We now summarize each chapter in more detail.

Chapter 2 challenges the de-facto mental model of uncertainties, and advocates that we instead think about *learned* versus *deduced* uncertainties. We present an online MC EM training process for the Laplace approximation (LA) that leads to better calibrated uncertainties. We present a novel Hessian approximation that enables LA to scale to large networks with high output resolution.

Chapter 3 presents the Mapillary Street-Level Sequences, a large dataset for place recognition. We find that some images do not contain enough visual information for accurate retrieval, which can lead to silent failures. We extend the Laplace approximation to metric learning, such that we can obtain stochastic embeddings.

Chapter 4 proposes a detector-agnostic method for quantifying the uncertainties of feature detectors. We find that this yields calibrated uncertainties for feature matching and that these uncertainties can improve downstream applications such as camera localization.

Chapter 5 explores two challenging problems in multiview stereo: how to represent dynamic scenes and how to incorporate learned 3D priors into the reconstruction process. In particular, we explore NeRF-based methods. We present an efficient planar decomposition that generalizes to any dimensional space, memory efficiently represents dynamic volumes at high resolutions and allows for fast training and rendering. We also propose to learn a probabilistic model of local 3D shapes that can be distilled into the reconstruction to improve the scene geometry and remove floaters. This leads to state-of-the-art reconstruction quality for casually captured NeRFs.

Chapter 6 presents a framework for disentangling and manipulating parts of a 3D scene, opening many applications across robotics and AR.

7.1 Future work

To conclude this thesis, there are many aspects of important future work which we would like to highlight. As is typical with any research, this thesis raises more questions than answers. Improvements to the individual algorithms have been discussed within the body of this work. However, at a high level, we would like to highlight the following themes for future research which are particularly exciting.

Uncertainties in Multiview Stereo. In Nerfbusters, we learn a probabilistic model for local 3D shapes, however, the focus was more on improving the geometry and removing artifacts than measuring the uncertainty of the reconstruction. We believe that quantifying the uncertainties in NeRF and implicit surface-based representations is an interesting direction with applications in active reconstruction [Pan+22], e.g. to guide a robot that is mapping an environment to where to collect images, or in physical simulations, where one wishes to reason about the uncertainty of a reconstruction. The Laplace approximation is an obvious candidate for quantifying the uncertainty of such implicit reconstruction. Very recently, Bayes Rays [Gol+23] showed that posthoc Laplace approximation for a NeRF yields a good measure of uncertainty. Using these uncertainties to improve the reconstruction, e.g. explore multiple hypotheses, or propagate the uncertainties to downstream applications will be very valuable.

System rather than parts. While this thesis has explored many of the core technologies of the reconstruction pipeline, it has explored each part of the system individually. This has led to a focus on output uncertainties. However, in practice, we believe input uncertainties are most interesting. These are uncertainties that are propagated through a system and can be used to avoid failures. We believe that combining the methods presented for each step into a single probabilistic 3D pipeline that propagates uncertainties throughout the pipeline would show the benefits of reasoning about uncertainties, reducing silent failures, and improving retrieval, localization, reconstruction, and manipulation.

Bibliography

[ALF21] Omri Avrahami, Dani Lischinski, and Ohad Fried. "Blended diffusion for text-driven editing of natural images." In: *arXiv preprint arXiv:2111.14818* (2021).

[Ara+16] Relja Arandjelović et al. "NetVLAD: CNN architecture for weakly supervised place recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pages 5297–5307.

[Bal+17] Vassileios Balntas et al. "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors." In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2017, pages 5173–5182.

[Bar+21] Jonathan T Barron et al. "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pages 5855–5864.

[Bar+22a] O León Barbed et al. "SuperPoint features in endoscopy." In: *Imaging Systems for GI Endoscopy, and Graphs in Biomedical Image Analysis: First MICCAI Workshop, ISGIE 2022, and Fourth MICCAI Workshop, GRAIL 2022, Held in Conjunction with MICCAI 2022, Singapore, September 18, 2022, Proceedings.* Springer. 2022, pages 45–55.

[Bar+22b] Jonathan T Barron et al. "Mip-nerf 360: Unbounded anti-aliased neural radiance fields." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pages 5470–5479.

[Ben+22] Sagie Benaim et al. "Volumetric Disentanglement for 3D Scene Manipulation." In: *arXiv preprint arXiv:2206.02776* (2022).

[Ben+23] Thoranna Bender et al. "Learning to Taste: A Multimodal Wine Dataset." In: *arXiv preprint arXiv:2308.16900* (2023).

[BM22] Axel Barroso-Laguna and Krystian Mikolajczyk. "Key. net: Keypoint detection by handcrafted and learned cnn filters revisited." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2022), pages 698–711.

[Boj+17] Piotr Bojanowski et al. *Optimizing the Latent Space of Generative Networks.* 2017. DOI: 10.48550/ARXIV.1707.05776. URL: https://arxiv.org/abs/1707.05776.

[BRB17] Aleksandar Botev, Hippolyt Ritter, and David Barber. "Practical Gauss-Newton optimisation for deep learning." In: *International Conference on Machine Learning (ICML)*. PMLR. 2017, pages 557–565.

[Cha+15] Angel X Chang et al. "Shapenet: An information-rich 3d model repository." In: *arXiv preprint arXiv:1512.03012* (2015).

[Cha+22] Eric R. Chan et al. "Efficient Geometry-aware 3D Generative Adversarial Networks." In: *CVPR*. 2022, pages 16102–16112. DOI: `10.1109/CVPR52688.2022.01565`.

[Che+22] Anpei Chen et al. "TensoRF: Tensorial Radiance Fields." In: *ECCV*. 2022.

[Chr+22] Peter Ebert Christensen et al. "Searching for Structure in Unfalsifiable Claims." In: *arXiv preprint arXiv:2209.00495* (2022).

[Dax+21] Erik Daxberger et al. "Laplace Redux - Effortless Bayesian Deep Learning." In: *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021).

[DCH88] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. "Volume rendering." In: *ACM Siggraph Computer Graphics* 22.4 (1988), pages 65–74.

[DD09] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural safety* 31.2 (2009), pages 105–112.

[Dep17] National Highway Traffic Safety Administration Department of Transportation. *PE 16-007. Technical report, U.S. Tesla Crash Preliminary Evaluation Report.* Technical report. 2017.

[Det+21] Nicki S. Detlefsen et al. *StochMan*. 2021. URL: `https://github.com/MachineLearningLifeScience/stochman/`.

[DKH20] Felix Dangel, Frederik Kunstner, and Philipp Hennig. "BackPACK: Packing more into Backprop." In: *International Conference on Learning Representations (ICLR)*. 2020.

[DL90] John Denker and Yann LeCun. "Transforming neural-net output levels to probability distributions." In: *Advances in Neural Information Processing Systems* (1990).

[DMR18] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description." In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pages 224–236.

[DSP20] Mihai Dusmanu, Johannes L Schönberger, and Marc Pollefeys. "Multi-view optimization of local feature geometry." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer. 2020, pages 670–686.

[Dus+19] Mihai Dusmanu et al. "D2-net: A trainable cnn for joint description and detection of local features." In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. 2019, pages 8092–8101.

[Fan+22] Jiemin Fang et al. "Fast Dynamic Radiance Fields with Time-Aware Neural Voxels." In: *SIGGRAPH Asia 2022 Conference Papers*. ACM, 2022. DOI: `10.1145/3550469.3555383`. URL: `https://doi.org/10.1145%2F3550469.3555383`.

[FC20] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences." In: *Minds and Machines* 30 (2020), pages 681–694.

[FH+15] Yasutaka Furukawa, Carlos Hernández, et al. "Multi-view stereo: A tutorial." In: *Foundations and Trends® in Computer Graphics and Vision* 9.1-2 (2015), pages 1–148.

[FH97] F Dan Foresee and Martin T Hagan. "Gauss-Newton approximation to Bayesian learning." In: *Proceedings of International Conference on Neural Networks (ICNN'97)*. Volume 3. IEEE. 1997, pages 1930–1935.

[Fri+22] Sara Fridovich-Keil et al. "Plenoxels: Radiance fields without neural networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pages 5501–5510.

[Fri+23] Sara Fridovich-Keil et al. "K-planes: Explicit radiance fields in space, time, and appearance." In: *arXiv preprint arXiv:2301.10241* (2023).

[Gan+22] Wanshui Gan et al. *V4D: Voxel for 4D Novel View Synthesis*. 2022. DOI: `10.48550/ARXIV.2205.14332`. URL: `https://arxiv.org/abs/2205.14332`.

[Gao+22] Hang Gao et al. "Monocular dynamic view synthesis: A reality check." In: *Advances in Neural Information Processing Systems*. 2022.

[GDS20] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. "Evaluating scalable bayesian deep learning methods for robust computer vision." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pages 318–319.

[Gei+13] Andreas Geiger et al. "Vision meets robotics: The kitti dataset." In: *The International Journal of Robotics Research* 32.11 (2013), pages 1231–1237.

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning." In: *international conference on machine learning*. PMLR. 2016, pages 1050–1059.

[Gol+23] Lily Goli et al. "Bayes' Rays: Uncertainty Quantification for Neural Radiance Fields." In: *arXiv preprint arXiv:2309.03185* (2023).

[Guo+22] Rui Guo et al. "GANmouflage: 3D Object Nondetection with Texture Fields." In: *arXiv preprint arXiv:2201.07202* (2022).

[He+16] Kaiming He et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pages 770–778.

[HG16] Dan Hendrycks and Kevin Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks." In: *arXiv preprint arXiv:1610.02136* (2016).

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." In: *Advances in Neural Information Processing Systems* 33 (2020), pages 6840–6851.

[HS+07] Wolfgang Härdle, Léopold Simar, et al. *Applied multivariate statistical analysis*. Volume 22007. Springer, 2007.

[HS+88] Chris Harris, Mike Stephens, et al. "A combined corner and edge detector." In: *Alvey vision conference*. Volume 15. Citeseer. 1988, pages 10–5244.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks." In: *Science* 313.5786 (2006), pages 504–507.

[Hua+07] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Technical report 07-49. University of Massachusetts, Amherst, October 2007.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[IKB21] Alexander Immer, Maciej Korzepa, and Matthias Bauer. "Improving predictions of Bayesian neural nets via local linearization." In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pages 703–711.

[Jen+14] Rasmus Jensen et al. "Large scale multi-view stereopsis evaluation." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2014, pages 406–413.

[Jin+21] Yuhe Jin et al. "Image Matching Across Wide Baselines: From Paper to Practice." In: *Int. J. Comput. Vis.* 129.2 (2021), pages 517–547. DOI: 10.1007/s11263-020-01385-0. URL: https://doi.org/10.1007/s11263-020-01385-0.

[JTA21] Ajay Jain, Matthew Tancik, and Pieter Abbeel. "Putting nerf on a diet: Semantically consistent few-shot view synthesis." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pages 5885–5894.

[Kan96] Kenichi Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. USA: Elsevier Science Inc., 1996. ISBN: 0444824278.

[KG17] Alex Kendall and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems* 30 (2017).

[KHB19] Frederik Kunstner, Philipp Hennig, and Lukas Balles. "Limitations of the empirical Fisher approximation for natural gradient descent." In: *Advances in neural information processing systems* 32 (2019).

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational Bayes." In: *arXiv preprint arXiv:1312.6114* (2013).

[LDS89] Yann LeCun, John Denker, and Sara Solla. "Optimal brain damage." In: *Advances in Neural Information Processing Systems* (1989).

[Li+22a] Tianye Li et al. "Neural 3D Video Synthesis from Multi-view Video." In: *CVPR*. 2022, pages 5511–5521. DOI: `10.1109/CVPR52688.2022.00544`.

[Li+22b] Zhengqi Li et al. "Infinitenature-zero: Learning perpetual view generation of natural scenes from single images." In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*. Springer. 2022, pages 515–534.

[Lin+21] Chen-Hsuan Lin et al. "Barf: Bundle-adjusting neural radiance fields." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pages 5741–5751.

[Liu+21] Andrew Liu et al. "Infinite nature: Perpetual view generation of natural scenes from a single image." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pages 14458–14467.

[LMF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "Epnp: An accurate o (n) solution to the pnp problem." In: *International journal of computer vision* 81.2 (2009), pages 155–166.

[Low+16] Stephanie Lowry et al. "Visual place recognition: A survey." In: *IEEE Transactions on Robotics* 32.1 (2016), pages 1–19.

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles." In: *Advances in Neural Information Processing Systems* 30 (2017).

[Luo+20] Zixin Luo et al. "Aslfeat: Learning local features of accurate shape and localization." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pages 6589–6598.

[Luo+21] Katie Luo et al. "Stay Positive: Non-Negative Image Synthesis for Augmented Reality." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pages 10050–10060.

[Mac92] David J. C. MacKay. "Bayesian Interpolation." In: *Neural Computation* 4.3 (May 1992), pages 415–447.

[Mar+21] Ricardo Martin-Brualla et al. “Nerf in the wild: Neural radiance fields for unconstrained photo collections.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pages 7210–7219.

[Max95] N. Max. “Optical models for direct volume rendering.” In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pages 99–108. DOI: `10.1109/2945.468400`.

[MBF09] Jochen Meidow, Christian Beder, and Wolfgang Förstner. “Reasoning with uncertain points, straight lines, and straight line segments in 2D.” In: *ISPRS Journal of Photogrammetry and Remote Sensing* 64.2 (2009), pages 125–139.

[MBL20] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. “A metric learning reality check.” In: *European Conference on Computer Vision*. Springer. 2020, pages 681–699.

[Mel+23] Luke Melas-Kyriazi et al. “RealFusion: 360 {\deg} Reconstruction of Any Object from a Single Image.” In: *arXiv preprint arXiv:2302.10663* (2023).

[MG15] James Martens and Roger B. Grosse. “Optimizing Neural Networks with Kronecker-factored Approximate Curvature.” In: *International Conference on Machine Learning (ICML)*. 2015.

[Mia+22] Marco Miani et al. “Laplacian Autoencoders for Learning Stochastic Representations.” In: *Advances in Neural Information Processing Systems*. 2022.

[Mil+19] Ben Mildenhall et al. “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines.” In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pages 1–14.

[Mil+20] Ben Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis.” In: *ECCV*. Springer. 2020, pages 405–421.

[MIS17] Shohei Mori, Sei Ikeda, and Hideo Saito. “A survey of diminished reality: Techniques for visually concealing, eliminating, and seeing through real objects.” In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), pages 1–14.

[ML14] Mehdi Mekni and Andre Lemieux. “Augmented reality: Applications, challenges and future trends.” In: *Applied computational science* 20 (2014), pages 205–214.

[Mül+22] Thomas Müller et al. “Instant neural graphics primitives with a multiresolution hash encoding.” In: *ACM Transactions on Graphics (ToG)* 41.4 (2022), pages 1–15.

[Naz+19] Kamyar Nazeri et al. “Edgeconnect: Generative image inpainting with adversarial edge learning.” In: *arXiv preprint arXiv:1901.00212* (2019).

[ND21] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved denoising diffusion probabilistic models.” In: *International Conference on Machine Learning*. PMLR. 2021, pages 8162–8171.

[Nic+21] Alex Nichol et al. “Glide: Towards photorealistic image generation and editing with text-guided diffusion models.” In: *arXiv preprint arXiv:2112.10741* (2021).

[Nie+22] Michael Niemeyer et al. “RegNeRF: Regularizing Neural Radiance Fields for View Synthesis From Sparse Inputs.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pages 5480–5490.

[Oh+18] Seong Joon Oh et al. “Modeling uncertainty with hedged instance embedding.” In: *arXiv preprint arXiv:1810.00319* (2018).

[Owe+14] Andrew Owens et al. “Camouflaging an object from many viewpoints.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pages 2782–2789.

[Özy+17] Onur Özyeşil et al. “A survey of structure from motion*.” In: *Acta Numerica* 26 (2017), pages 305–364.

[Pal99] Stephen E Palmer. *Vision science: Photons to phenomenology*. MIT press, 1999.

[Pan+22] Xuran Pan et al. “Activenerf: Learning where to see with uncertainty estimation.” In: *European Conference on Computer Vision*. Springer. 2022, pages 230–246.

[Pen+20] Songyou Peng et al. “Convolutional Occupancy Networks.” In: *ECCV*. Edited by Andrea Vedaldi et al. Lecture Notes in Computer Science. 2020, pages 523–540. DOI: `10.1007/978-3-030-58580-8\_31`.

[Poo+22] Ben Poole et al. “Dreamfusion: Text-to-3d using 2d diffusion.” In: *arXiv preprint arXiv:2209.14988* (2022).

[Pum+21] Albert Pumarola et al. “D-NeRF: Neural Radiance Fields for Dynamic Scenes.” In: *CVPR*. 2021.

[Rad+21] Alec Radford et al. “Learning transferable visual models from natural language supervision.” In: *International Conference on Machine Learning*. PMLR. 2021, pages 8748–8763.

[RBB18] Hippolyt Ritter, Aleksandar Botev, and David Barber. “A scalable Laplace approximation for neural networks.” In: *International Conference on Learning Representations (ICLR)*. Volume 6. 2018.

[Rei+21] Jeremy Reizenstein et al. “Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction.” In: *International Conference on Computer Vision*. 2021.

[Rev+19] Jerome Revaud et al. “R2d2: Reliable and repeatable detector and descriptor.” In: *Advances in neural information processing systems* 32 (2019).

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation.” In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pages 234–241.

[Rom+21] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: `2112.10752 [cs.CV]`.

[RTC18] Filip Radenović, Giorgos Tolias, and Ondřej Chum. “Fine-tuning CNN image retrieval with no human annotation.” In: *IEEE transactions on pattern analysis and machine intelligence* 41.7 (2018), pages 1655–1668.

[Sab+23] Sara Sabour et al. “RobustNeRF: Ignoring Distractors with Robust Losses.” In: *arXiv preprint arXiv:2302.00833* (2023).

[Sar+19] Paul-Edouard Sarlin et al. “From coarse to fine: Robust hierarchical localization at large scale.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pages 12716–12725.

[Sch+22] Pola Schwöbel et al. “Probabilistic spatial transformer networks.” In: *Uncertainty in Artificial Intelligence*. PMLR. 2022, pages 1749–1759.

[SE19] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution.” In: *Advances in neural information processing systems* 32 (2019).

[SJ19] Yichun Shi and Anil K Jain. “Probabilistic face embeddings.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pages 6902–6911.

[SMS18] Javier Sánchez, Nelson Monzón, and Agustín Salgado De La Nuez. “An analysis and implementation of the harris corner detector.” In: *Image Processing On Line* (2018).

[Soh+15] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics.” In: *International Conference on Machine Learning*. PMLR. 2015, pages 2256–2265.

[SSC22] Cheng Sun, Min Sun, and Hwann-Tzong Chen. “Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction.” In: *CVPR*. 2022.

[Stu+12] Jürgen Sturm et al. “A benchmark for the evaluation of RGB-D SLAM systems.” In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pages 573–580.

[Tah+19a] Ahmed Taha et al. “Exploring uncertainty in conditional multi-modal retrieval systems.” In: *arXiv preprint arXiv:1901.07702* (2019).

[Tah+19b] Ahmed Taha et al. “Unsupervised data uncertainty learning in visual retrieval systems.” In: *arXiv preprint arXiv:1902.02586* (2019).

[Tan+21] Matthew Tancik et al. “Learned Initializations for Optimizing Coordinate-Based Neural Representations.” In: *CVPR*. Computer Vision Foundation / IEEE, 2021, pages 2846–2855. DOI: `10.1109/CVPR46437.2021.00287`.

[Tan+23] Matthew Tancik et al. “Nerfstudio: A Modular Framework for Neural Radiance Field Development.” In: *arXiv preprint arXiv:2302.04264* (2023).

[TFT20] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. “DISK: Learning local features with policy gradient.” In: *Advances in Neural Information Processing Systems* 33 (2020), pages 14254–14265.

[Tri+00] Bill Triggs et al. “Bundle adjustment—a modern synthesis.” In: *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*. Springer. 2000, pages 298–372.

[TWC23] Javier Tirado-Garín, Frederik Warburg, and Javier Civera. “DAC: Detector-Agnostic Spatial Covariances for Deep Local Features.” In: *arXiv preprint arXiv:2305.12250* (2023).

[Vak+21] Alexander Vakhitov et al. “Uncertainty-aware camera pose estimation from points and lines.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pages 4659–4668.

[Val+22] Andrea Vallone et al. “Danish airs and grounds: A dataset for aerial-to-street-level place recognition and localization.” In: *IEEE Robotics and Automation Letters* 7.4 (2022), pages 9207–9214.

[Ver+15] Yannick Verdie et al. “Tilde: A temporally invariant learned detector.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pages 5279–5288.

[VM22] Matias Valdenegro-Toro and Daniel Saromo Mori. “A deeper look into aleatoric and epistemic uncertainty disentanglement.” In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. 2022, pages 1508–1516.

[Wah+11] Catherine Wah et al. *Caltech-UCSD Birds 200*. 2011.

[Wan+04] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity.” In: *IEEE TIP* 13.4 (2004), pages 600–612. DOI: `10.1109/TIP.2003.819861`.

[Wan+21] Zirui Wang et al. “NeRF−: Neural radiance fields without known camera parameters.” In: *arXiv preprint arXiv:2102.07064* (2021).

[Wan+22a] Feng Wang et al. *Mixed Neural Voxels for Fast Multi-view Video Synthesis.* 2022. DOI: `10.48550/ARXIV.2212.00190`. URL: `https://arxiv.org/abs/2212.00190`.

[Wan+22b] Haochen Wang et al. "Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation." In: *arXiv preprint arXiv:2212.00774* (2022).

[War+20] Frederik Warburg et al. "Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June 2020.

[War+21] Frederik Warburg et al. "Bayesian triplet loss: Uncertainty quantification in image retrieval." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pages 12158–12168.

[War+22] Frederik Warburg et al. "Self-Supervised Depth Completion for Active Stereo." In: *IEEE Robotics and Automation Letters* 7.2 (2022), pages 3475–3482.

[War+23a] Frederik Warburg et al. "Bayesian metric learning for uncertainty quantification in image retrieval." In: *arXiv preprint arXiv:2302.01332* (2023).

[War+23b] Frederik Warburg et al. "Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs." In: *arXiv preprint arXiv:2304.10532* (2023).

[Wim+23] Lisa Wimmer et al. "Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures?" In: *Uncertainty in Artificial Intelligence.* PMLR. 2023, pages 2282–2292.

[Wiz+21] Suttisak Wizadwongsa et al. *NeX: Real-time View Synthesis with Neural Basis Expansion.* 2021. arXiv: `2103.05606 [cs.CV]`.

[WM23] Frederik Warburg and Marco Miani. "Pytorch-laplace." In: *GitHub. Note: https://github.com/FrederikWarburg/pytorch-laplace* (2023).

[WRL22] Frederik Warburg, Michael Ramamonjisoa, and Manuel López-Antequera. "SparseFormer: Attention-based Depth Completion Network." In: *arXiv preprint arXiv:2206.04557* (2022).

[WT23] Jamie Wynn and Daniyar Turmukhambetov. "DiffusioNeRF: Regularizing Neural Radiance Fields with Denoising Diffusion Models." In: *arxiv.* 2023.

[Xu+22] Kuan Xu et al. "AirVO: An Illumination-Robust Point-Line Visual Odometry." In: *arXiv preprint arXiv:2212.07595* (2022).

[Yi+16] Kwang Moo Yi et al. "Lift: Learned invariant feature transform." In: *European conference on computer vision.* Springer. 2016, pages 467–483.

[Yu+19] Jiahui Yu et al. "Free-form image inpainting with gated convolution." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pages 4471–4480.

[Yu+21a] Alex Yu et al. "pixelnerf: Neural radiance fields from one or few images." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021, pages 4578–4587.

[Yu+21b] Alex Yu et al. "Plenoctrees for real-time rendering of neural radiance fields." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pages 5752–5761.

[Zaf+19] Mubariz Zaffar et al. "Levelling the playing field: A comprehensive comparison of visual place recognition approaches under changing conditions." In: *arXiv preprint arXiv:1903.09107* (2019).

[Zep+23] Kilian Zepf et al. "Laplacian Segmentation Networks: Improved Epistemic Uncertainty from Spatial Aleatoric Uncertainty." In: *arXiv preprint arXiv:2303.13123* (2023).

[ZF18] Yinda Zhang and Thomas Funkhouser. "Deep depth completion of a single rgb-d image." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pages 175–185.

[Zho+22] Qunjie Zhou et al. "Is Geometry Enough for Matching in Visual Localization?" In: *European Conference on Computer Vision.* Springer. 2022, pages 407–425.

[ZT22] Zhizhuo Zhou and Shubham Tulsiani. "SparseFusion: Distilling View-conditioned Diffusion for 3D Reconstruction." In: *arXiv preprint arXiv:2212.00792* (2022).