Analysis by Synthesis for Modern Computer Vision

Nathaniel E. Chodosh

CMU-RI-TR-24-55

August 2024

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee

Simon Lucey,	Co-Chair
Deva Ramanan,	Co-Chair
David Held	
Noah Snavely	(Cornell Tech)

Thesis proposal submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

© Nate Chodosh, 2024

Keywords: Compressed Sensing, Analysis by Synthesis, 3D Reconstruction, 3D Computer Vision

Abstract

Image denoising, depth completion, scene flow, and dynamic 3D reconstruction are all examples of recovery problems: the estimation of multidimensional signals from corrupted or partial measurements. This thesis examines these problems from the classic analysis-by-synthesis perspective, where a signal model is used to propose hypotheses, which are then compared to observations. This paradigm has fallen out of favor with the rise of feed-forward neural networks, but we claim that analysis by synthesis still has much to offer. Specifically, we argue it gives us a general framework for combining modern learning-based approaches with knowledge of forward models and intuitive priors.

First, we will discuss the typical feed-forward setting where one has a dataset of paired measurements and clean signals. In this setting, we show how embedding an analysis by synthesis optimization within the learning process can help us enforce constraints and generalize to new forward models. Second, we will focus on the self-supervised setting in the context of scene-flow estimation, a task where we only have indirect measurements (sequences of point clouds) of the signal of interest (motion). In this case, we will see how a test-time optimization can create a learning target for a feed-forward network that can then be scaled to large unlabeled datasets. Finally, we will examine a problem of estimating multiple signals simultaneously from measurements: the recovery of geometry and motion from sequences of point clouds. Here, instead of embedding an optimization into the learning process, we do the reverse. We show how a global analysis-by-synthesis objective can be broken down into components appropriate for off-the-shelf models. In each of these problems, we will see that analysis by synthesis offers us a powerful and flexible paradigm for structuring our approaches and injecting learning in the right places.

Contents

1 Introduction **9**

1.1 Differentiable Optimization for Supervised Recovery Problems (ACCV 2019, CVPR 2020) . . . 11

1.2 A Student–Teacher Paradigm for an Un-Supervised Recovery Problem (WACV 2023, ICLR 2024) . . . 12

1.3 A Decompositional Approach for a Multi-Signal Un-Supervised Recovery Problem (In Submission) . . . 12

2 Supervised Recovery Problems **14**

2.1 Introduction . . . 14

2.2 Preliminary . . . 16

2.3 Deep Convolutional Compressed Sensing . . . 18

2.4 Experiments . . . 21

2.5 Other Inverse Problems . . . 27

3 Un-Supervised Recovery Problems **36**

3.1 Introduction . . . 36

3.2 Related Work . . . 38

3.3 Benchmark Issues . . . 39

3.4 Baseline . . . 43

3.5 Evaluation . . . 47

3.6 Conclusion . . . 51

3.7 Student/Teacher Distillation . . . 51

3.8 Experiments . . . 54

4 Towards Multi-Signal Recovery **60**

4.1 Introduction . . . 60

4.2 Related Work . . . 63

4.3 Problem Statement . . . 63

4.4 Objective . . . 65

4.5 Experimental Setup . . . 70

4.6 Qualitative Results . . . 71

4.7 Quantitative Results 71
4.8 Conclusion 73

5 Conclusions **75**

Acknowledgements

First and foremost, I want to thank Simon and Deva; without their support, this thesis could never have happened. Simon taught me so much in the first years of my PhD, from the basics of writing a paper or giving a talk to the core research skills of identifying interesting angles on important problems. Above all, though, he always believed in me, even when I didn't. I am forever grateful to Deva for coming in during one of those tough times, integrating me into the wonderful community in his lab, coaching me through the academic job search, and being an incredible mentor in the back half of my thesis work.

Second, I want to thank the people who made it possible for me to get to graduate school in the first place: my parents, my partner, my family, my friends. Completing a doctorate is an unusual achievement, but my parents made it seem like the most natural thing in the world and gave me the skills required to do it. The most special thank you, of course, goes to my partner Emma: without her love and support, I would have given up a long time ago. My family and friends form the foundation of my life, never failing to offer encouragement, compassion, or just a cheap place to live in Queens.

Finally, I want to thank the community I've been lucky enough to find here at CMU. To my roommate Michael: I will always remember our clashes on the squash court and the countless activities and outings you organized. To the students I first met here, Chen-Hsuan, Chaoyang, Ming-Fang, Ben, Peiyun, and Leo, I owe all of the basic skills one needs to survive at CMU and my first research community. Last but not least, I want to thank Neehar, Tarasha, Anish, Jason, Haithem, Kyle, and all of Deva's other students who have been so kind and welcoming since I became a late-stage addition to the group.

List of Figures

1.1 The focus of this thesis is the comparison and combination of two methods for recovery problems in computer vision: (a) analysis by synthesis and (b) feed-forward. In the analysis by synthesis paradigm, signals are reconstructed by proposing hypotheses, which are passed through a model of the measurement process and then compared to the actual measurements. The feed-forward approach instead learns the entire mapping from measurements to predicted signals by drawing on a large dataset of examples (blue box). The critical differences between these approaches are that the analysis by synthesis approach allows for easy integration of forward models whereas the feed-forward approach allows for easy scaling to large datasets. 10

2.1 The bicyclist and bollards can barely be seen in the input map but a clearly represented in the output. Our method also accurately reconstructs very thin objects such as the sign post. On the right it can be seen that our method enforces that its prediction should match the input points while the SparseConvNet systematically underestimates the depth. 15

2.2 Results on the KITTI benchmark for varying levels of input sparsity. The keep probability represents the probability that any particular LiDAR sample is retained. We demonstrate robustness to reasonable changes in input sparsity, outperforming both baselines up to a 50% reduction in the number of input points. 23

2.3 Results of selected methods on the KITTI benchmarks for varying training set sizes. Our method performs well with training sizes ranging from 100-86k but still benefits from larger training sizes. 23

2.4 Results on the depth completion benchmark for different numbers of ADMM iterations. The total error is shown in blue while the red line shows the error on just those points given as input. The dotted lines show the same metrics but for the SparseConvNet of Uhrig *et al.* [14]. 24

2.5 Selected visual results form the KITTI benchmark. From top to bottom: RGB Image, Ground truth, input LiDAR points, Predicted depth. 26

2.6 Top row, from left to right: an image from the Berkeley Segmentation Database compressed using JPEG, the same image degrade by our linear approximation, the original image. Bottom row, a zoom of the top images to show the blocking artifacts. 27

2.7 Our baseline CNN architecture. It is based on the JPEG Artifact Removal architecture of Zheng *et al.* . Due to the high dimensionality of the block-diagonal transformation we first do a patch wise encoding step. The patch-wise encoder operates on a single 4x4 patch and corresponding block of the transformation, it encodes it to a 64 channel 4x4 image which concatenated with the original image and then fed through a standard CNN. The residual blocks are of the same form as Zheng *et al.* 28

2.8 The results of the synthetic expperiment on BSD-500. For each value of α we train our CNN 5 times to convergence and then report the average of the top three validation results. In contrast we only train our model one time at $\alpha = 0.5$ which explains why the performance of our model peaks there since the hyper-parameters have been learned for that value. 29

2.9 Some example artifact removal results from BSD-500, zoomed in to highlight the detail. From left to right: Degraded input image, target image, our result 31

2.10 A comparison of our method and that of Zheng *et al.* on the JPEG artifact removal task. Note that as the degradation becomes more sever, that is the blocks of the measurement matrix become less diagonal, we see the same kind of fall off in performance as we did in the synthetic experiment. 31

2.11 From top to bottom: image degraded at QF 5, our method's output, the method of Sulam *et al.* , the target image. The poor performance of the method of Sulam *et al.* can be explained by the fact that these methods are very sensitive to the choice of parameters. The dictionary is learned with one set of parameters but there is no clear way to modify them once we introduce the JPEG degradation. The table reports RMSE for our method and Sulam *et al.* at quality factors 5 and 25 32

2.12 Some example trajectories and our reconstruction results 33

2.13 Performance vs Number of layers for our proposed Trajectory CNN . 34

2.14 Qualitative results on BSD500. The captions contain RMSE, PSNR, and SSIM measures for each image 34

3.1 Recent self-supervised scene flow methods[113, 83, 79, 70] typically focus on performance on stereoKITTI[88]. We call attention to several problematic aspects of its semi-synthetic construction. When evaluated on real-world datasets (Waymo, Argoverse, NuScenes) we find a **negative correlation with stereoKITTI performance**. For each method and LiDAR dataset, we plot a point corresponding to the self-reported end-point error on stereoKITTI versus the end-point error on dynamic points of that method trained on real data. For each dataset, we plot the best-fit line to visualize the correlation. 37

3.2 The correlation between performance on stereoKITTI and performance on real datasets as a function of how much those datasets violate the one-to-one correspondence assumption present in stereoKITTI. The more a dataset violates this assumption (higher chamfer distance due to sparser pointclouds), the worse the correlation with stereoKITTI performance. 39

3.3 The performance of ICP and NSFP versus the ratio of dynamic points in each example. The green region indicates the ratio found in real data and the red region indicates the ratio found in the stereoKITTI dataset. The unrealistic dynamic ratio of stereoKITTI causes ICP to appear to perform worse than it does on real-world data such as Argoverse. 40

3.4 An analysis of the motion profile of points found in various autonomous driving datasets. The leftmost column is the percentage of points belonging to any tracked object. The right three columns show those points separated by their speed once ego-motion has been removed. All three datasets have a very low rate of dynamic foreground points, in contrast to the typical stereoKITTI benchmark. . . 41

3.5 Comparison of sampling patterns from KITTI-SF (**left**) versus the corresponding real LiDAR scan (**right**). KITTI-SF uses dense CAD models for foreground objects, which makes it easier for learning-based methods to find them among the sparse LiDAR background points. 42

3.6 An example ground truth segmentation from KITTI-SF (**left**) compared with (**right**) an example prediction from our local model. Note how the model can even identify parked vs moving cars since only moving vehicles have the dense sampling pattern. 42

3.7 Our proposed pipeline for LiDAR scene flow based largely on classical pre- and post-processing. 44

3.8 Comparison between NSFP with our proposed pre and post processing steps (**left**) and standard NSFP (**right**). In the (**bottom**) views points are color coded by EPE. The (**top**) detail views show the first and second frames aligned by the predicted flow. NSFP struggles to represent both foreground and background motion. We find that first using ICP to remove ego-motion greatly improves the estimates on dynamic points. 44

3.9 An example of our ground removal technique on a Waymo [80] scene with a non-planar ground. In all images, color coding indicates the height of each point. From left to right: the input point cloud, the result of thresholding, our result. 44

3.10 Optimizing for the standard nearest neighbor self-supervised loss can cause mis-predictions in the presence of strong occlusions. Here, the bed of the truck is collapsed into the cab by NSFP (**left**). Our RANSAC non-rigid refinement step can fix this type of error (**right**). 46

3.11 The *Scene Flow via Distillation* (SFvD) framework, which describes a new class of scene flow methods that produce high quality, human label-free flow at the speed of feedforward networks. 52

3.12 Empirical scaling laws for ZeroFlow. We report Argoverse 2 validation split Threeway EPE as a percentage of the Argoverse 2 *train* split used, on a $\log_{10}$-$\log_{10}$ scale, trained to convergence. Threeway EPE performance of ZeroFlow scales logarithmically with the amount of training data. 57

3.13 Normalized frame birds-eye-view heatmaps of endpoint residuals for Chamfer Distance, as well as the outputs for NSFP and Chodosh on moving points (points with ground truth speed above 0.5m/s). Perfect predictions would produce a single central dot. Top row shows the frequency on a $\log_{10}$ color scale, bottom row shows the frequency on an absolute color scale. Qualitatively, methods with better quantitative results have tighter residual distributions. 58

4.1 Surface reconstruction of a dynamic sequence from NuScenes. Given ego-pose and bounding box annotations, a naive approach would aggregate background and object points into common reference frames and then run a point-to-surface reconstruction algorithm. Even human annotations are not accurate enough for this simple approach (**top left**). Instead, we design an optimization that refines both the ego and object poses, yielding high-quality reconstructions (**top right**). The input LiDAR sweep is plotted with red and blue spheres, red for background points and blue for dynamic. The naive approach also fails due to rolling shutter effects on fast-moving vehicles (**bottom left**) which we correct for (**bottom right**) 61

4.2 LiDAR returns are often grouped and processed in 360-degree sweeps, but most sensors have a continuous "shutter" as they rotate. Our framework can model this continuous shutter for any combination of LiDAR sensors or scanning patterns. We find that this modeling has a large impact on the quality of reconstructed objects. Here, we visualize the set of points within a sweep, which are captured simultaneously (green lines) for NuScenes (**top**) and Argoverse (**bottom**). Argoverse has two LiDAR sensors spinning 180 degrees out of phase, leading to two sets of points being captured at each instant. 62
4.3 An example dense depth map produced by our method 64
4.4 A high-level overview of our method when used with sparse ground truth annotations. We take the annotated LiDAR frames that make use of interpolation and off-the-shelf LiDAR odometry to initialize object and ego poses for all frames. Our global optimization makes use of coordinate descent to update the geometry and motion alternatingly. When using the output of an object tracker as input, we omit the interpolation step, as the tracks cover all the input frames. 64
4.5 (**Left**) A LiDAR sweep where each point has been colored according to which laser it belongs to (hue) and the time within the sweep it was acquired (lighter is earlier, darker is later). A moving car is passing the ego-vehicle on the left and is captured at both the start and end of the sweep (**top right**), leading to distortion (the driver-side window is captured twice in different locations). Accounting for this distortion by modeling the object motion is key to the quality of our reconstructions (**bottom right**). 68
4.6 Accurate object poses and accounting for intra-sweep motion are critical for high-quality reconstructions. (**Left**) shows the reconstruction with neither refined poses nor object-motion compensation, (**middle**) shows the reconstruction with refined poses but without object-motion compensation, and (**right**) shows the result of combining both. 68
4.7 (**Left)** Each column shows NuScenes object reconstructions using ground truth poses compared to (**right**) ours. 70
4.8 (**Left)** Argoverse object reconstructions using ground truth poses compared to (**right**) ours. 71
4.9 (**Left)** Map reconstruction on Argoverse 2.0 using *ground truth* poses compared to (**right**) ours. 73
4.10 (**Left)** Map reconstructions using odometry poses compared to (**right**) ours. Ground-truth ego poses produce even worse results since NuScenes does not align poses in the z (height) dimension. . . 74

List of Tables

2.1 Validation error of various methods on the KITTI Depth Completion benchmark. All results except for SparseConvNet and Ma's are taken as reported from [14]. Our method outperforms all previous state-of-the-art depth only completion methods (Middle) as well as those that use RGB images for guidance (Top). 22

2.2 Results on BSD500 for reconstruction trained and tested at a specific quality factor, the numbers in each cell are RMSE, PSNR, and SSIM. Best results are in bold face, and results which are close to the best are underlined 27

2.3 Results of multi-layer CSC (ML-CSC) on the trajectory reconstruction problem. We report Normalized RMSE in addition to RMSE, to better compare sequences of different scales. Please see [53] for a description of this metric. 33

3.1 Quantitative results on Argoverse 2. Our baseline predicts the motion of dynamic objects by 30% over [70], despite that method using ground truth foreground masks for training. We also see from the static background EPE that ICP outperforms learning-based methods at predicting ego-motion. 47

3.2 Quantitative results on NuScenes. Our baseline halves the EPE on dynamic objects when compared to the next best method (NSFP). Since our baseline also uses NSFP as its backbone, this indicates that pre- and post-processing steps can have an enormous impact. Again we also see that ICP has the best ego-motion prediction. 47

3.3 Quantitative results on Waymo. Our baseline outperforms all tested methods but notably in this instance [70] performs better than ICP at predicting the background ego-motion. 48

3.4 The results of our baseline on NuScenes as evaluated by Baur *et al.* [57]. U and S refer to the unsupervised and fully-supervised versions of their method. Our dataless baseline outperforms all self-supervised methods and even achieves comparable and superior results to the fully supervised network. 48

3.5 Results of our method on lidarKITTI w/ ground [70]. We outperform all existing methods without using any training data. 48
3.6 Quantitative results on the Argoverse 2 Sensor validation split using the evaluation protocol from [119]. The methods used in this paper, shown in the first two blocks of the table, are trained and evaluated on point clouds within a 102.4m × 102.4m area centered around the ego vehicle (the settings for the *Argoverse 2 Self-Supervised Scene Flow Challenge*) . However, following the protocol of [119], all methods report error on points in the 70m × 70m area centered around the ego vehicle. Runtimes are collected on an NVIDIA V100 with a batch size of 1 [135]. FastFlow3D, ZeroFlow 1X, and ZeroFlow 3X have identical feedforward architectures and thus share the same real-time runtime; FastFlow3D XL, ZeroFlow XL 1X, and ZeroFlow XL 3X have identical feedforward architectures and thus share the same runtime. Methods with an * have performance averaged over 3 training runs. Underlined methods require human supervision. . . . 55
3.7 Quantitative results on Waymo Open using the evaluation protocol from [119]. Runtimes are scaled to approximate the performance on a V100 [125]. Both FastFlow3D and ZeroFlow 1X have identical feedforward architectures and thus share the same runtime. Underlined methods require human supervision. 56
3.8 Comparison between ZeroFlow trained on Argoverse 2 Sensor dataset versus the more diverse, unlabeled Argoverse 2 LiDAR subset described in Section 3.8.1. Diverse training datasets result in non-trivial performance improvements. 58
3.9 Comparison between ZeroFlow trained on Argoverse 2 using NSFP pseudo-labels, ZeroFlow using [119] pseudo-labels, and ZeroFlow using TruncatedChamfer. Methods with an * have performance averaged over 3 training runs. The minor quality improvement of Chodosh pseudo-labels does not lead to a meaningful difference in performance, while the significant degradation of TruncatedChamfer leads to significantly worse performance. 59

4.1 Surface quality evaluation on NuScenes, measured by comparing the LiDAR points to their closest points on the reconstructed surfaces. . 72
4.2 Pose accuracy evaluation on NuScenes (using NuScene's default ATE metric), measured by comparing the bounding box locations predicted by our method to held-out ground truth labels provided at 2Hz. We compare our method to linearly interpolating the poses as is commonly done to create scene-flow labels [57]. 73

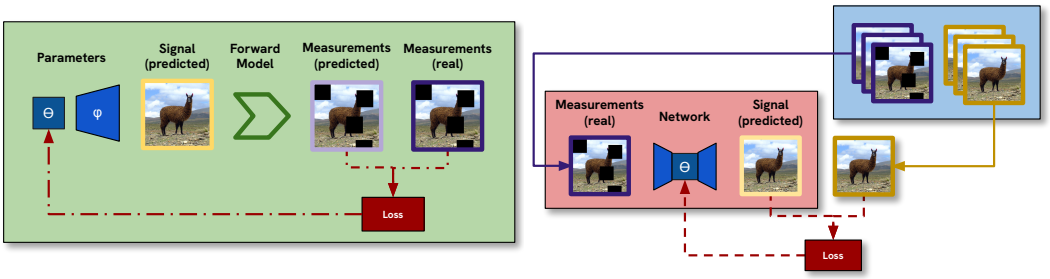
Chapter 1

Introduction

A recurring problem in computer vision and robotics is the recovery of an underdetermined multidimensional signal from corrupted or partial measurements. We call such tasks recovery problems. The most fundamental multidimensional signal in computer vision is RGB images, but we will also consider other signals, such as 3D scene geometry and motion. To give a concrete example, some classic recovery problems aim to produce clean, high-resolution images from noisy, blurry, or low-resolution inputs. These problems contain the fundamental structure we will use to analyze all tasks discussed in this thesis. Specifically, they have (1) a multidimensional signal to be recovered, (2) a known forward model that produces (3) observed measurements. Historically, one of the main paradigms for approaching these types of recovery problems has been Analysis by Synthesis.

In the context of recovery problems, analysis by synthesis is a paradigm where an internal model of the signal space is combined with the forward model to produce a hypothesis of the original. The quality of this hypothesis can be measured by passing it through the forward model and comparing the results to the observations. This naturally leads to an optimization-based inference where the reconstruction parameters are tuned to recreate the observations. Total variation denoising is an illustrative example of this approach. In this example, the forward model is additive Gaussian noise, and the signal model is that images with small gradients are more likely than those with large gradients. Combining these gives the classical optimization objective. These approaches were popular since they provided an intuitive language for encoding knowledge of sensing processes and signal priors. However, the increased availability of large datasets and the success of deep learning have steered most research away from this paradigm. Instead, many works have used these large datasets to train neural networks to predict recovered signals from measurements directly.

This new paradigm, which we will call the "feed-forward approach", has been remarkably successful. Feed-forward networks have proven to be effective at learning the classic recovery problems, such as super-resolution and deblurring, and more



(a) (b)

Figure 1.1: The focus of this thesis is the comparison and combination of two methods for recovery problems in computer vision: (a) analysis by synthesis and (b) feed-forward. In the analysis by synthesis paradigm, signals are reconstructed by proposing hypotheses, which are passed through a model of the measurement process and then compared to the actual measurements. The feed-forward approach instead learns the entire mapping from measurements to predicted signals by drawing on a large dataset of examples (blue box). The critical differences between these approaches are that the analysis by synthesis approach allows for easy integration of forward models whereas the feed-forward approach allows for easy scaling to large datasets.

complex problems, such as the prediction of dense geometry from partial point clouds or even dense geometry directly from images. Key to all of these successes has been the ability of feed-forward networks to utilize vast amounts of data, eliminating the need for handcrafted priors. Given this "bitter lesson," it is unclear what use the classic analysis-by-synthesis approach has. This thesis aims to revisit this classic perspective in light of advances in machine learning.

The claim is that when faced with a recovery problem, analysis by synthesis gives us a general framework for combining modern learning-based approaches with knowledge of forward models and intuitive priors. We will examine three different settings for recovery problems. First, we will discuss the typical feed-forward setting where one has a dataset of paired measured and clean signals. In this setting, we show how embedding an analysis by synthesis optimization within the learning process can help us enforce constraints and generalize to new forward models. Second, we will focus on the self-supervised setting in the context of scene-flow estimation, a task where we only have indirect measurements (sequences of point clouds) of the signal of interest (motion). In this case, we will see how a test-time optimization can create a learning target for a feed-forward network that can be scaled to large unlabeled datasets. Finally, we will examine a problem of estimating multiple signals simultaneously from measurements: the recovery of geometry and motion from sequences of point clouds. Here, instead of embedding an optimization

into the learning process, we do the reverse. We show how a global analysis-by-synthesis objective can be broken down into components appropriate off-the-shelf learned methods can solve. In each of these problems, we will see that analysis by synthesis offers us a powerful and flexible paradigm for structuring our approaches and injecting learning in the right places.

1.1 Differentiable Optimization for Supervised Recovery Problems (ACCV 2019, CVPR 2020)

We will begin with the setting most amenable to the feed-forward approach. We call this setting a "supervised recovery problem," where one has access to a large dataset of pairs of original and measured signals, which can then be used to train a neural network to map measurements to recovered signals. The typical feed-forward formulation of these problems optimizes the weights of an encoder-decoder network to minimize the sum of reconstruction errors over the whole dataset. A typical objective looks like

$$\min_{\Theta} \sum_{(\mathbf{x}_i, \mathbf{y}_i)} \mathcal{L}\left(\mathbf{y}_i, f(\mathbf{x}_i; \Theta)\right) \tag{1.1}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ are pairs of observations and clean signals and $\mathcal{L}$ is an appropriate reconstruction loss. We will specifically consider the reconstruction problems of super-resolution for LiDAR, removing compression artifacts from RGB images, and lifting 2D point tracks to 3D trajectories. All of these problems have existing feed-forward approaches similar to eq. (1.1). This straightforward approach can be successful, but in contrast to analysis by synthesis, it lacks a sensible way of incorporating knowledge of the forward model.

Instead of directly learning the mapping from measurements to signals, we draw on the classical compressed sensing approach of learning a sparse signal model which can then be inverted to solve each instance of the recovery problem. Sparsity is a property of many real-world signals which says that a clean signal $\mathbf{y}$ can be written as a linear combination of a small number of basis elements. The distinction between a sparsity and low-rank model is that each signal $\mathbf{y}$ may be supported by a different small subset of a large set of possible basis vectors. That is, the signal space is *locally* low rank. Sparsity has a rich history of theoretical and practical results. Still, one of the classical weaknesses is that finding the set of possible basis vectors can be challenging.

The insight we take from the feed-forward approach is that as long as the inference process can be differentiated, we can use the given dataset to implicitly learn the parameters that lead to the best reconstructions. This naturally leads to a learning objective similar to eq. (1.1) but with the feed-forward network replaced with an inner optimization that performs sparse code recovery. We then reduce this bi-level optimization to a single-level one through loop unrolling. In LiDAR densification,

restoration of compressed images, and the recovery of 3D motion from 2D tracks, we show that this bilevel optimization can effectively learn models that give good signal reconstructions and generalize to different measurement models at test time.

1.2 A Student–Teacher Paradigm for an Un-Supervised Recovery Problem (WACV 2023, ICLR 2024)

The supervised setting is powerful, but in many cases, we do not have a large dataset of paired clean and measured signals. Some important signals are extremely difficult or impossible to measure directly. The example we are concerned with is scene flow, the 3D motion of all points in a scene, specifically in the context of autonomous driving. Autonomous vehicles use LiDAR to make sparse measurements of the scene geometry but still need to estimate the motion of other objects. Using human labelers to produce a dataset of ground truth motion labels is prohibitive, and as a result, there is significant interest in label-free or self-supervised approaches.

In this setting, we will see another benefit of the analysis-by-synthesis approach: it is often very easy to incorporate intuitive priors into our models. This gives us the surprising result that in this situation where we have no examples of the clean scene flow signal, incorporating human priors can be more effective than the feed-forward advantage of scaling to large datasets. To see this, we comprehensively evaluated top feed-forward scene flow models across multiple autonomous driving settings. Our analysis of the evaluation revealed that a data-free test-time optimization that minimized a simple motion model over the same self-supervised loss as the feed-forward methods could significantly outperform them. Thus, we concluded that these self-supervised methods did not learn anything from the dataset that could not be learned from a single example.

We show that rather than trying to glue these priors to a feed-forward network, we can adopt a student-teacher paradigm that lets us transfer the information in our analysis by synthesis model to a feed-forward network. To do this, we run our optimization over each example in our dataset to produce pseudo-labels, which are then distilled into a student network through supervised learning on the pseudo-labels. The student network eventually outperforms our teacher optimization when scaled to a large dataset, demonstrating that this method can achieve the best of both worlds of learning from human priors and large-scale data.

1.3 A Decompositional Approach for a Multi-Signal Un-Supervised Recovery Problem (In Submission)

In the previous settings, we were concerned with reconstructing one signal from a set of measurements. In the wild, however, we often are concerned with multiple signals that, when combined, give rise to a sequence of measurements. In the final

portion of this thesis, we tackle one such problem: the reconstruction of geometry and motion from a sequence of LiDAR measurements.

Once again, we will see that analysis by synthesis gives a powerful perspective for relating measurements to underlying signals. In this instance, we explore a different mechanism for infusing this perspective with modern learning techniques. We use analysis-by-synthesis to combine motion, geometry, and LiDAR measurement models into a global optimization, which can be broken down into sub-components appropriate for off-the-shelf learned models. We find that an accurate model of the measurement process is critical to achieving high-quality reconstruction, and our analysis by synthesis makes it clear how that model should be incorporated.

Chapter 2

Supervised Recovery Problems

2.1 Introduction

In recent years 3D information has become an important component of robotic sensing. Usually this information is presented in 2.5D as a depth map, either measured directly using LiDAR or computed using stereo correspondence. Since LiDAR and stereo techniques yield few samples relative to modern image sensors, it has become desirable to convert sparse depth measurements into high resolution depth maps as shown in Figure 2.1.

Recent works [10, 14] have directly applied deep networks to depth completion from sparse measurements. However, common network architectures have two drawbacks when applied to this task: 1) They implicitly pose depth completion as finding a mapping from sparse depth maps to dense ones, instead of as finding a depth map that is consistent with the sparse input. This essentially throws away information and we observe that feed forward networks do not learn to propagate the input points through to the output. Qualitative evidence of this can be seen in Figure (2.1). 2) Common networks are sensitive to the sparsity of the input since they treat all pixels equally, regardless of whether or not they represent samples or missing input. Special CNN networks have been designed to address this problem, but they still do not express the constraints given by the input [14]. In this paper we address both of these issues with a novel deep recurrent autoencoder architecture, which internally optimizes its depth prediction with respect to both sparsity and input constraints. To do this, we have taken inspiration from Compressed sensing (CS) which provides a natural framework for this problem. Formally, CS is concerned with recovering signals from incomplete measurements by enforcing that signals be sparse when measured in an appropriate basis. This basis takes the form of an overcomplete matrix which maps sparse representations to observed signals.

The choice of dictionary is crucial for recovering the signal efficiently, especially when the dimensionality is high. For high resolution imagery data, such as depth maps, multi-layer convolutional sparse coding (CSC) [12] is effective as it explicitly

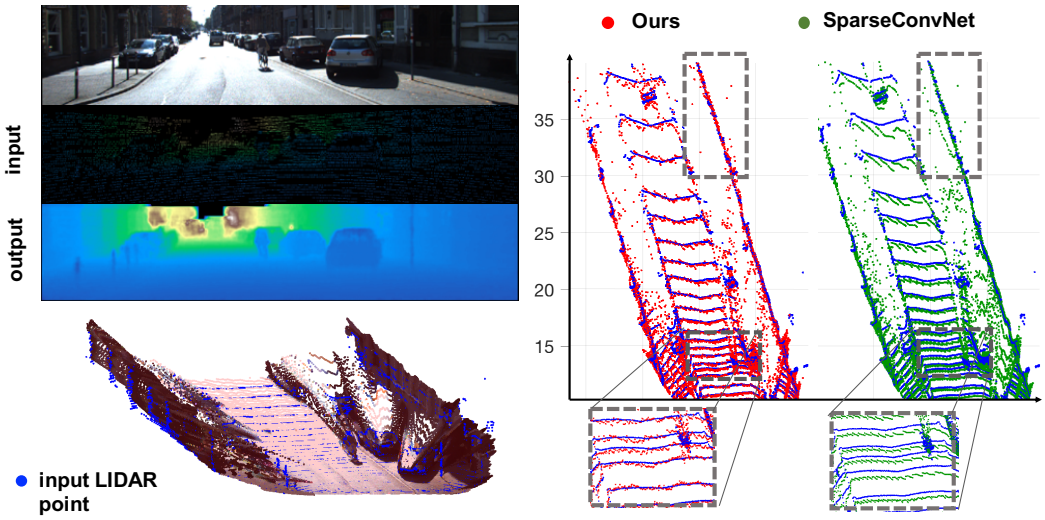


Figure 2.1: The bicyclist and bollards can barely be seen in the input map but a clearly represented in the output. Our method also accurately reconstructs very thin objects such as the sign post. On the right it can be seen that our method enforces that its prediction should match the input points while the SparseConvNet systematically underestimates the depth.

models local interactions through the convolution operator with tractable computational and model complexity. However, none of the existing multi-layer convolutional sparse coding algorithms are designed for learning from sparse ground truth data. This is reflected by the fact that recent works [7, 8] applying CS to depth completion are restricted to using single-level, hand crafted dictionaries. CS has also fallen out of fashion since the existing algorithms have difficult to interpret hyper-parameters, and often do not achieve good performance without careful tuning of these parameters.

Recent developments in the formal analysis of deep learning have shown that convolutional neural networks and convolutional sparse coding are closely related. Specifically it has been shown that CNNs with ReLU activation functions are carrying out a specific form of the layered thresholding algorithm for CSC. Layered thresholding is a simple algorithm for solving multi-layered convolutional sparse coding (ML-CSC) problems, which can be effective when there is little noise and the coherence of the dictionary is high. Motivated by the work of Murdock *et al.* [13], in this paper we propose a network architecture which encodes a more sophisticated algorithm for ML-CSC. Encoding the ML-CSC objective in a deep network allows us to learn the dictionaries and parameters together in an end to end fashion. We show that by better approximating this objective, we can out perform all published results on the KITTI depth completion benchmark while using far fewer

parameters and layers. Furthermore, this work builds on the Alternating Direction Neural Network (ADNN) framework of Murdock *et al.* which gives theoretical insight into deep learning and we believe is a promising new area of research.

To summarize, the main contributions of this paper are:

1. We frame an end-to-end multi-layer dictionary learning algorithm as a neural network. This allows us to effectively learn dictionaries and hyper-parameters from a large dataset. In comparison, existing CS algorithms either use hand crafted dictionaries, separately learned multi-level dictionaries, or are inapplicable to incomplete training data [15], as is our case.

2. Our method allows for explicit encoding of the constraints from the input sparse depth. Current deep learning approaches [10] simply feed in a sparse depth map and rely solely on data to teach the network to identify which inputs represent missing data. Some recent models [14] explicitly include masks to achieve sparsity invariance, but none have a guaranteed way of encoding that the input is a noise corrupted subset of the desired output. In contrast our method directly optimizes the predicted map with respect to the input.

3. Our method demonstrates state-of-the-art performance with much fewer parameters compared to deep networks. In fact, using only two layers of dictionaries and 1600 parameters, our method already substantially outperforms modern deep networks which use more than 20 layers and over 3 million parameters [10]. As a result of having fewer parameters, our approach trains faster and requires less data.

2.2 Preliminary

2.2.1 Compressed sensing

Compressed sensing concerns the problem of recovering a signal from a small set of measurements. In our case, we're interested in reconstructing the depth map $\mathbf{d}$ with full resolution from the sparse depth map $\mathbf{d}_s$ produced by LiDAR. To achieve this, certain prior knowledge of the signal is required. The most widely used prior assumption is that the signal can be reconstructed with a sparse linear combination of basis elements from an over-complete dictionary $\mathbf{W}$. This gives an optimization problem similar to sparse coding:

$$\min_{\mathbf{z}} \|\mathbf{MWz} - \mathbf{d}_s\| + b \|\mathbf{z}\|_0 , \tag{2.1}$$

where $\mathbf{z}$ is the code, $\mathbf{Wz}$ produces our predicted depth map, and $\mathbf{M}$ is a diagonal matrix with 0 and 1s on its diagonal. It's used to mask out the unmeasured portions of the signal, such that the reconstruction error is only applied to the pixels which have been measured.

The key question to apply CS in Eq. 2.1 is: 1) For high dimensional signals such as the depth map, how to design the dictionary such that it encourages uniqueness of the code while still being computationally feasible; 2) How to learn the dictionary to get best reconstruction accuracy. In Sec. 2.3, we are going to show that the dictionary can be factored into a structure equivalent to performing multi-layer convolution, and that we can unroll the optimization of Eq. 2.1 into a network similar to a deep recurrent neural network. This allows us to learn the dictionary together with other hyper-parameters (*e.g.* b) through end-to-end training.

2.2.2 Deep Component Analysis

Equation (2.1) can be generalized to multi-layered sparse coding in which one seeks a very high level sparse representation $\mathbf{z}_\ell$ such that $\mathbf{d} = \mathbf{W}_1\mathbf{W}_2 \dots \mathbf{W}_{\ell-1}\mathbf{z}_\ell$ and each intermediate product $\mathbf{z}_i = \mathbf{W}_i\mathbf{W}_{i+1} \dots \mathbf{W}_{\ell-1}\mathbf{z}_\ell$ is also sparse. This formulation makes using a large effective dictionary computationally tractable, and when the dictionaries have a convolutional structure it allows for increased receptive fields while keeping the number of parameters manageable. This is further generalized to Deep Component Analysis (DeepCA) by the recent work of Murdock *et al.* which replaces the ℓ_0 loss with arbitrary sparsity-encouraging penalties. The DeepCA objective function is stated in [13] as:

$$\min_{\{\mathbf{z}_i\}} \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{z}_{i-1} - \mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{z}_i), \tag{2.2}$$

where the Φ_j are sparsity encouraging regularizers. Previous work has shown that the specific choice of $\Phi(\mathbf{x}) = I(\mathbf{x} > 0) + b\|\mathbf{x}\|_1$ yields optimization algorithms very similar to a feed-forward neural network with Relu activation functions. By using the ADMM algorithm to solve equation (2.18), Murdock *et al.* create Alternating Direction Neural Networks, a generalization of feed forward neural networks which internally solve optimization problems with the form of (2.18). Alternating Direction Neural Networks (ADNNs) perform the optimization in a fully differentiable manner and cast the activation functions of each layer as the proximal operators of penalty function Φ_i of that layer. This allows for learning the dictionaries W_i and parameters b through gradient descent and back propagation with respect to an arbitrary loss function on the sparse codes. To mirror neural networks, Murdock *et al.* apply various loss functions to the highest level of codes, which take the place of the output layer in traditional NNs. In the following sections we will show how ADNNs can be adapted to the depth completion problem within the framework of compressed sensing.

2.3 Deep Convolutional Compressed Sensing

2.3.1 Inference

Directly applying compressed sensing to the DeepCA objective gives

$$\min_{\{\mathbf{z}_i\}} \frac{1}{2} \|\mathbf{d}_s - \mathbf{M}\mathbf{W}_1\mathbf{z}_1\|_2^2 + \sum_{i=2}^{\ell} \frac{1}{2} \|\mathbf{z}_{i-1} - \mathbf{W}_i\mathbf{z}_i\|_2^2 + \sum_{i=1}^{\ell} \Phi_i(\mathbf{z}_i), \tag{2.3}$$

where $\mathbf{d}_s$ is the input sparse depth map. However, if we take the $\mathbf{W}_i$ to have a convolutional structure then an element $\mathbf{z}_1$ will not be recovered if its spatial support contains no valid depth samples. Thus, extracting the higher level codes is itself a missing data problem and can be written the same way. This gives the full Deep Convolutional Compressed Sensing objective:

$$\min_{\{\mathbf{z}_i | i>0\}} \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{M}_{i-1}\mathbf{z}_{i-1} - \mathbf{M}_{i-1}\mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{z}_i). \tag{2.4}$$

Here, to simplify notation, we merge the depth reconstruction cost (left term in Eq. 2.3) and the reconstruction cost of the codes together, with $\mathbf{z}_0 = \mathbf{d}_s$ and $\mathbf{M}_0$ denotes the mask $\mathbf{M}$ used in (2.3) . Each $\mathbf{M}_i$ is a mask encoding which elements of $\mathbf{z}_i$ had any valid inputs in their spatial support. In practice computing $\mathbf{M}_i$ is done with a maxpooling operation with the same stride and kernel size as the convolution represented by $\mathbf{W}_{i+1}^T$.

We solve (2.4) using the ADMM algorithm, which introduces auxiliary variables $\mathbf{y}_i$ that we constrain to be equal to the codes $\mathbf{z}_i$ as below:

$$\begin{aligned} \min_{\{\mathbf{y}_i, \mathbf{z}_i | i>0\}} & \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{M}_{i-1}\mathbf{y}_{i-1} - \mathbf{M}_{i-1}\mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{y}_i) \\ \text{s.t.} \quad & \mathbf{z}_i = \mathbf{y}_i. \end{aligned} \tag{2.5}$$

Here, we again refer the input sparse depth $\mathbf{d}_s$ as $\mathbf{y}_0$. With this, the augmented Lagrangian of (2.5) with dual variables $\boldsymbol{\lambda}$ and a quadratic penalty weight ρ is:

$$L_\rho(\mathbf{z}, \mathbf{y}, \boldsymbol{\lambda}) = \sum_{i=1}^{\ell} \frac{1}{2} \|\mathbf{M}_{i-1}\mathbf{y}_{i-1} - \mathbf{M}_{i-1}\mathbf{W}_i\mathbf{z}_i\|_2^2 + \Phi_i(\mathbf{y}_i) + \boldsymbol{\lambda}_i^T(\mathbf{z}_i - \mathbf{y}_i) + \frac{\rho}{2} \|\mathbf{z}_i - \mathbf{y}_i\|_2^2 . \tag{2.6}$$

The ADMM algorithm then minimizes L_ρ over each variable in turn, while keeping all others fixed. Following Murdock *et al.* we will incrementally update each layer instead of first solving for all $\mathbf{z}_i$ followed by all $\mathbf{y}_i$. They show this order leads to faster convergence. The ADMM updates for each variable are as follows:

1. At each iteration $t+1$, $\mathbf{z}_i$ is first updated by minimizing L_ρ with the associated auxiliary variable $\mathbf{y}_i$ from the previous iteration, and $\mathbf{z}_{i-1}$ from the current iteration fixed:

$$\begin{aligned}\mathbf{z}_i^{[t+1]} &= \arg\min_{\mathbf{z}_i} L_\rho(\mathbf{z}_i, \mathbf{y}_{i-1}^{[t+1]}, \mathbf{y}_i^{[t]}, \boldsymbol{\lambda}_i^{[t]}) \\ &= (\mathbf{W}_i^T \mathbf{M}_{i-1}^T \mathbf{M}_{i-1} \mathbf{W}_i + \rho \mathbf{I})^{-1} (\mathbf{W}_i^T \mathbf{M}_{i-1}^T \mathbf{M}_{i-1} \mathbf{W} \mathbf{y}_{i-1}^{[t+1]} + \rho \mathbf{y}_i^{[t]} - \boldsymbol{\lambda}^{[t]}).\end{aligned} \tag{2.7}$$

This gives a fully differentiable update of $\mathbf{z}_i$ but the matrix inversion is computationally expensive, especially since W_i is in practice very large. To deal with this problem we make the approximation that $\mathbf{W}_i$ is a Parseval tight frame [13], that is we assume $\mathbf{W}_i \mathbf{W}_i^T = \mathbf{I}$. In addition to being common practice in autoencoders with tied weights, this assumption is also made by Murdock *et al.* and has previously been explicitly enforced in deep neural networks [23]. We can then use the binomial matrix identity to rewrite the $\mathbf{z}_i$ update as:

$$\mathbf{z}^{[t+1]} = \tilde{\mathbf{y}}_i^{[t]} + \frac{1}{1+\rho} \mathbf{W}_i^T \mathbf{M}_{i-1}^T (\mathbf{M}_{i-1} \mathbf{y}_{i-1}^{[t+1]} - \mathbf{M}_{i-1} \mathbf{W}_i \tilde{\mathbf{y}}_i^{[t]}), \tag{2.8}$$

where $\tilde{\mathbf{y}}_i^{[t]} \triangleq \mathbf{y}_i^{[t]} - \frac{1}{\rho}$.

2. Similarly, the update rule for the auxiliary variables $\mathbf{y}_i$ is:

$$\begin{aligned}\mathbf{y}_i^{[t+1]} &= \arg\min_{\mathbf{y}_i} L_\rho(\mathbf{z}_i^{[t+1]}, \mathbf{z}_{i+1}^{[t]}, \mathbf{y}_i, \boldsymbol{\lambda}_i^{[t]}) \\ &= \phi_i \left(\frac{1}{1+\rho} \mathbf{W}_{i+1} \mathbf{z}_{i+1}^{[t]} + \frac{\rho}{1+\rho} (\mathbf{z}_i^{[t+1]} + \frac{\boldsymbol{\lambda}_i^{[t]}}{\rho}) \right) \\ \mathbf{y}_\ell &= \phi_i \left(\mathbf{z}_i^{[t]} + \frac{\boldsymbol{\lambda}_i^{[t]}}{\rho} \right).\end{aligned} \tag{2.9}$$

Here ϕ_i is the proximal operator associated with the penalty function Φ_i. For appropriate choices of Φ_i, ϕ_i is differentiable and can be computed efficiently. With this in mind, we choose $\Phi_i(\mathbf{x}) = I(\mathbf{x} > 0) + b \|\mathbf{x}\|_1$ so that $\phi_i(\mathbf{x}) = \text{ReLU}(\mathbf{x} - \frac{b}{\rho})$.

3. Finally the dual variable $\boldsymbol{\lambda}_i$ is updated by:

$$\boldsymbol{\lambda}_i^{[t+1]} = \boldsymbol{\lambda}_i^{[t]} + \rho(\mathbf{z}_i^{[t+1]} - \mathbf{y}_i^{[t+1]}). \tag{2.10}$$

The full procedure is detailed in algorithm (1). As shown in above, all the operations used in the ADMM iteration are differentiable, and can be implemented with deep learning layers *e.g.* convolution, convolution transpose, and ReLU. We

Algorithm 1: Deep Convolutional Compressed Sensing

Input : model parameters $\mathbf{W}_i, b_i$, iterations T, sparse depth $\mathbf{y}_0 = \mathbf{d}_s$, mask $\mathbf{M}$

Output: sparse codes $\mathbf{z}_i^{[T]}$, predicted depth $\mathbf{d}_{\text{pred}}$

for $i \leftarrow 1$ **to** ℓ **do**

  $\mathbf{z}_i^{[0]} \leftarrow \mathbf{W}_i^T \mathbf{M}_{i-1}^T \mathbf{y}_{i-1}$;

  $\mathbf{y}_i^{[0]} \leftarrow ReLU(\mathbf{z}_i^{[0]} - b/\rho)$;

  $\boldsymbol{\lambda}_i^{[0]} \leftarrow 0$;

for $t \leftarrow 1$ **to** T **do**

  for $i \leftarrow 1$ **to** ℓ **do**

    Update $\mathbf{z}_i^{[t]}$ using equation (2.8);

    Update $\mathbf{y}_i^{[t]}$ using equation (2.9);

    Update $\boldsymbol{\lambda}_i^{[t]}$ using equation (2.10);

  Predict $\mathbf{d}_{\text{pred}}$ using equation (2.11);

unroll the ADMM iteration for a constant number of iterations T, and output our optimized code $\mathbf{z}_\ell$ for the last layer. We can then extract our prediction of the depth map by applying the effective dictionary to the high level code z_ℓ as shown in equation (2.11). This is different from the standard decoder portion of a deep autoencoder, where the nonlinear activations are applied in between each convolution. Our approach does not require this since the internal optimization of z_ℓ enforces equality constraints between layers, which is not the case for conventional autoencoders. We choose to reconstruct the depth from z_ℓ instead of a lower layer because its elements have the largest receptive field and therefore z_ℓ will have the fewest number of missing entries.

$$\mathbf{d}_{\text{pred}} = \mathbf{W}_1 \mathbf{W}_2 \dots \mathbf{W}_\ell \mathbf{z}_\ell \tag{2.11}$$

2.3.2 Learning

With the ADMM update unrolled to T iterations as described above, the entire inference procedure can be thought of as a single differentiable function:

$$\mathbf{d}_{\text{pred}} = f_{\text{DCCS}}^{[T]}(\mathbf{M}, \mathbf{d}_s; \{\mathbf{W}_i, b_i\}) \tag{2.12}$$

Thus the dictionaries $\mathbf{W}_i$ and the bias term b_i which are the parameters for $f_{\text{DCCS}}^{[T]}$ can be learned through stochastic gradient descent over a suitable loss function. Using the standard sum of squared loss error, dictionary learning is formed as min-

imizing the depth reconstruction error $\mathcal{L}_{\text{reconstruct}}$:

$$\min_{\{\mathbf{W}_i, b_i\}} \sum_{n=1}^{N} \left\| \mathbf{d}_{\text{gt}}^{(n)} - \mathbf{M}'^{(n)} \mathbf{d}_{\text{pred}}^{(n)}) \right\| \tag{2.13}$$

Where $\mathbf{d}_{\text{gt}}^{(n)}$ is the ground truth depth map of the n_{th} training example. We allow the ground truth depth map to have missing value by using mask $\mathbf{M}'^{(n)}$ to segment out the invalid pixels in the ground truth depth map.

In practice we found that due to the sparsity of the training data, the depth maps our method predicted were rather noisy. To fix this issue we included the well known anisotropic total variation loss (TV-L1) when training to encourage smoothness of the predicted depth map. Note that this change has no significant impact on the quantitative error metrics, but produces more visually pleasing outputs. The total loss is then given by summation of the depth reconstruction loss and the TV-L1 smoothness loss, with hyper-parameter α to control the weighting for the smoothness penalty:

$$\mathcal{L} = \mathcal{L}_{reconstruct} + \alpha \mathcal{L}_{\text{TV-L1}}. \tag{2.14}$$

We empirically determined that $\alpha = 0.1$ produces the best results.

2.4 Experiments

2.4.1 Implementation Details

We implemented three variants of algorithm (1) for the cases $\ell = 1, 2, 3$. For the single layer case we let $\mathbf{W}_1^T$ be a 11x11 convolution with striding of 2 and 8 filters. For $\ell = 2$ we let $\mathbf{W}_1^T$ be an 11x11 convolution with 8 filters and W_1^T be a 7x7 convolution with 16 filters. Finally for the $\ell = 3$ case: $\mathbf{W}_1^T$ is an 11x11 convolution with 8 filters, $\mathbf{W}_2^T$ is a 5x5 convolution with 16 filters, and $\mathbf{W}_3^T$ is a 3x3 convolution with 32 filters. For both $\ell = 2$ and $\ell = 3$, all convolutions have striding of 2. For the single layer case we learned the dictionaries with the number of iterations set to 5 and then at test time increased the number of iterations to 20. For the two and three layer cases the number of iterations was fixed at train and test time to 10 except in section 2.4.4 where the number of test and training iterations is varied. All training was done with the ADAM optimizer with the standard parameters: learning rate = 0.001,$\beta_1 = 0.9$, β_2 =0.999, $\epsilon = 10^{-8}$.

2.4.1.1 Error Metrics

For evaluation on the KITTI benchmark we use the conventional error metrics [14, 4], *e.g.* root mean square error (RMSE), mean absolute error (MAE), mean absolute relative error (MRE). We also use the percentage of inliers metric, δ_i which counts

	RMSE (m)	MAE (m)	MRE	$\delta_1 < 1.01$	$\delta_2 < 1.01^2$	$\delta_3 < 1.01^3$
Bilateral NN[17]	4.19	1.09	-	-	-	-
SGDU[18]	2.5	0.72	-	-	-	-
Fast Bilateral Solver[19]	1.98	0.65	-	-	-	-
TGVL[20]	4.85	0.59	-	-	-	-
Closest Depth Pooling	2.77	0.94	-	-	-	-
Nadaraya Watson[21, 22]	2.99	0.74	-	-	-	-
ConvNet	2.97	0.78	-	-	-	-
ConvNet + mask	2.24	0.79	-	-	-	-
SparseConvNet[14]	1.82	0.58	0.035	0.33	0.65	0.82
Ma & Karaman[10]	1.68	0.70	0.039	0.21	0.41	0.59
Ours 1 Layer	2.77	0.83	0.054	0.3	0.47	0.59
Ours 2 Layers	1.45	0.47	0.028	0.41	0.68	0.8
Ours 3 Layers	**1.35**	**0.43**	**0.024**	**0.48**	**0.73**	**0.83**

Table 2.1: Validation error of various methods on the KITTI Depth Completion benchmark. All results except for SparseConvNet and Ma's are taken as reported from [14]. Our method outperforms all previous state-of-the-art depth only completion methods (Middle) as well as those that use RGB images for guidance (Top).

the percent of predictions whose relative error is within a threshold raised to the power i. Here, we use smaller thresholds (1.01^i) compared to the more widely used ones (1.5^i) in oder to compare differences in performance under tighter metrics.

2.4.2 KITTI Depth Completion Benchmark

We evaluate our method on the new KITTI Depth Completion Benchmark [14] instead of directly comparing against the LiDAR measurements from the raw KITTI dataset. The raw LiDAR points given in KITTI are corrupted by noise, motion of the vehicle during sampling, image rectification artifacts, and accounts to only 4% of the total number of pixels in the image. Thus it's not ideal for evaluating depth completion systems. Instead, the benchmark proposed in [14] resolved these issues by accumulating LiDAR measurements from nearby frames in the video sequences, and automatically removing accumulated LiDAR points that deviate too far from the points reconstructed by semi-global matching. This provides quality ground truth and effectively simulates the main application of interest: recovering dense depth from a single LiDAR sweep.

In Table 2.1, we form a close comparison against the very deep Sparse-to-Dense network (Ma & Karaman [10]) and the Sparsity Invariant CNN (SparseConvNet [14]) which are the current sate-of-the-art deep learning-based method.

The Sparse-to-Dense network uses a similar deep network architecture as those used for single shot depth prediction – with Resnet-18 as the encoder and up-projection blocks for the decoder. While the Sparse-to-Dense network is able to achieve good RMSE, it falls behind the SparseConvNet on MAE. We believe that this is because the deeper network can better estimate the average depth of a region but is unable to predict fine detail, leading to a higher MAE. By comparison, our method is able to both estimate the correct average depth and reconstruct fine detail due to its ability to directly optimize the prediction with respect to the input. Most notably our method outperforms all of the existing methods by a wide margin,

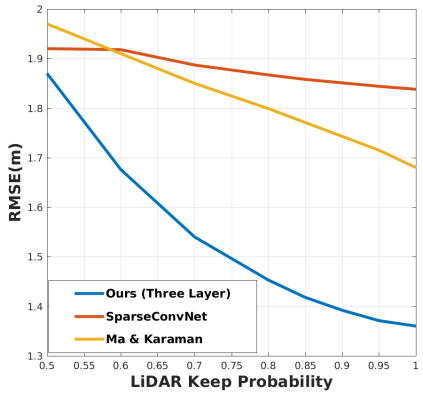


Figure 2.2: Results on the KITTI benchmark for varying levels of input sparsity. The keep probability represents the probability that any particular LiDAR sample is retained. We demonstrate robustness to reasonable changes in input sparsity, outperforming both baselines up to a 50% reduction in the number of input points.

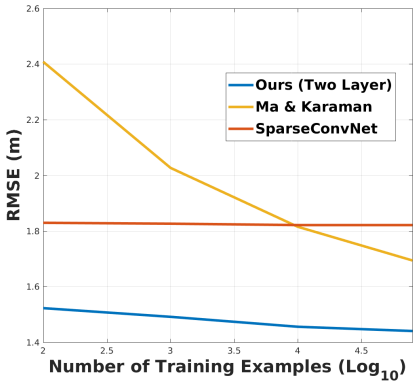


Figure 2.3: Results of selected methods on the KITTI benchmarks for varying training set sizes. Our method performs well with training sizes ranging from 100-86k but still benefits from larger training sizes.

including those that use RGB images and those that use orders of magnitude more parameters than our method.

2.4.2.1 Varying Sparsity Levels

Uhrig *et al.* show that their Sparsity Invariant CNNs are very robust to a mismatch between the training and testing levels of sparsity. While we do not see a practical use for disparities as large as those tested in [14], we do believe that depth completion systems should perform well under reasonable sparsity changes. To this end we adjusted the level of input sparsity in the KITTI benchmark by dropping input samples with probability p, for various values of p. The results of this experiment are shown in Figure (2.2). While it is clear that our method does not achieve the level of sparsity invariance of the SparseConvNet, it still outperforms both baseline results even when the only 50% of the input samples are kept.

2.4.3 Effect of Amount of Training Data

Modern deep learning models typically have tens of thousands to millions of parameters and therefore require enormous training sets to achieve good performance. This is in fact the motivation for the KITTI depth completion dataset, since previ-

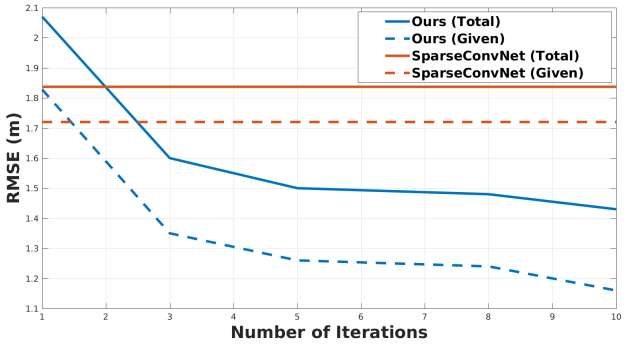


Figure 2.4: Results on the depth completion benchmark for different numbers of ADMM iterations. The total error is shown in blue while the red line shows the error on just those points given as input. The dotted lines show the same metrics but for the SparseConvNet of Uhrig *et al.* [14].

ous benchmarks did not have enough data to train deep networks. In this section we investigate the dependence on the amount of training data on the performance of our method in comparison with a standard deep network and the sparsity invariant variety.

Figure (2.3) shows the results of evaluating these models on the 1k manually selected validation depth maps after training on varying subsets of the 86k training maps. Our method outperforms both baselines for all training sizes. As expected Ma & Karaman's method fails to generalize well when trained on a small dataset since the model has 3.4M parameters but performs well once trained on the full dataset. It is interesting to observe that the method of Uhrig *et al.* does not gain any performance from training on more data. As a result it is ultimately out performed by the deep network which does not take sparsity into account. Our method is able to perform comparably to the sparsity invariant network with only 100 training examples but does increase in performance when given more data, validating the need for learning layered sparse coding dictionaries from large training sets.

2.4.4 Effect of Iterative Optimization

In this section we demonstrate that the success of our approach comes from its ability to refine depth estimates over multiple iterations. Applying a feed forward neural network to this problem frames it as finding a mapping from sparse LiDAR points to true depth maps. This is a reasonable approach but it doesn't utilize all of the available information, specifically it doesn't encode the relationship that input samples are a subset of the output that has been corrupted by noise. In contrast, our approach of phrasing depth completion as a compressed sensing missing data problem directly expresses that relationship. By solving this problem in an iterative

fashion our network that is able to find depth maps that are both consistent with the input constraints and have sparse representations.

The importance of iterative optimization is shown in Figure (2.4) where we examine the performance of our method as a function of the number of ADMM iterations it uses. It is clear that with few iterations our network fails to enforce the constraints and performs comparably to the SparseConvNet. This is also consistent with Murdock *et al.* 's observation that a feed forward network resembles a single iteration of an ADNN. As we increase the number of iterations our method is able to better optimize its prediction and gains a substantial performance boost.

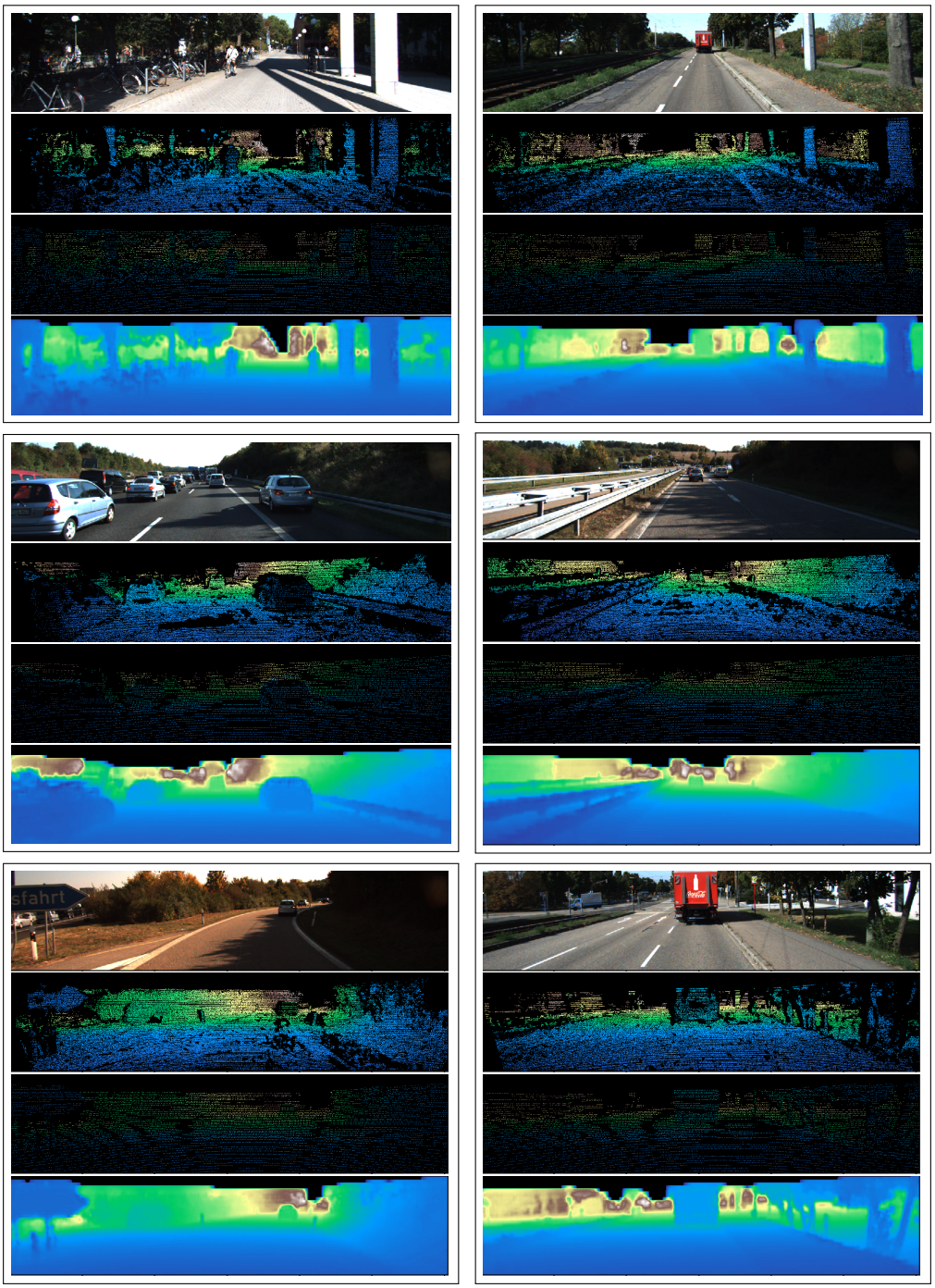


Figure 2.5: Selected visual results form the KITTI benchmark. From top to bottom: RGB Image, Ground truth, input LiDAR points, Predicted depth.

F-footnotesize.png

Dataset	Quality	Fu*et al.* [34]	Zheng*et al.* [52]	Ours
BSD500	5	0.0466 | 26.8 | 0.768	0.0381 | 28.5 | 0.853	**0.0360** | **29.05** | **0.860**
	10	0.0336 | 29.7 | 0.845	**0.0269** | **32.1** | **0.912**	0.0273 | 31.6 | 0.892
	20	0.0276 | 31.5 | 0.879	0.0249 | 32.4 | 0.906	**0.0218** | **33.5** | **0.918**
Parameters ($\times 10^5$)		3.74	106.6	.782

Table 2.2: Results on BSD500 for reconstruction trained and tested at a specific quality factor, the numbers in each cell are RMSE, PSNR, and SSIM. Best results are in bold face, and results which are close to the best are underlined

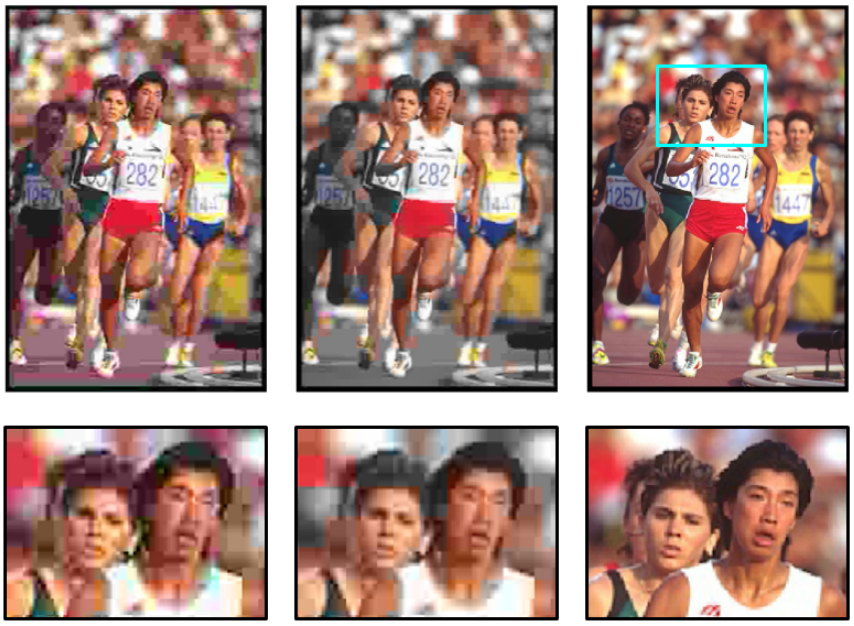


Figure 2.6: Top row, from left to right: an image from the Berkeley Segmentation Database compressed using JPEG, the same image degrade by our linear approximation, the original image. Bottom row, a zoom of the top images to show the blocking artifacts.

2.5 Other Inverse Problems

2.5.1 Diagonal vs Block Diagonal Inverse Problems

In this section we perform a synthetic experiment to show how the structure of the measurement matrix greatly affects the performance of a naive CNN approach to solving inverse problems. First we provide some intuition as to why this might be the

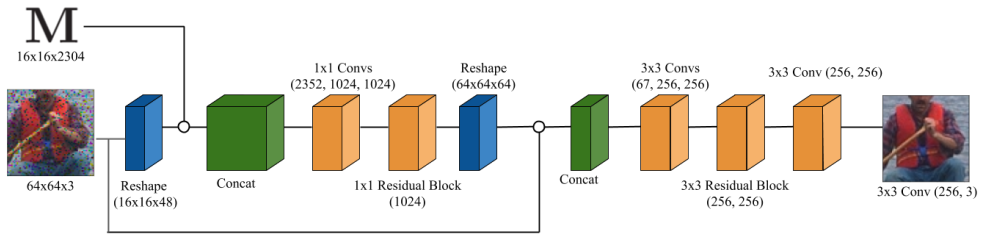


Figure 2.7: Our baseline CNN architecture. It is based on the JPEG Artifact Removal architecture of Zheng *et al.* . Due to the high dimensionality of the block-diagonal transformation we first do a patch wise encoding step. The patch-wise encoder operates on a single 4x4 patch and corresponding block of the transformation, it encodes it to a 64 channel 4x4 image which concatenated with the original image and then fed through a standard CNN. The residual blocks are of the same form as Zheng *et al.*

case. The structure of a CNN encodes the belief that the input signal has a convolutional structure, that is, an individual sample is highly correlated with its neighbors. When an image is degraded by missing data or blurring, these correlations are modified, but still persist. In these cases one would expect a CNN to perform well at reconstructing the original signal and many results confirm this [26, 51, 44, 45]. In these tasks the corresponding measurement matrix is diagonal or convolutional.

Now consider an inverse problem where the measurement matrix is block diagonal. For simplicity we will assume that the blocks are arranged such that each 4x4 patch of the image is multiplied by an arbitrary linear transformation. We will refer to this problem as block recovery. From the perspective of solving an inverse problem, this situation is identical to inpainting or deblurring, but we would expect it to be much more difficult for a CNN.

To test this hypothesis we performed a synthetic experiment where we interpolate smoothly between inpainting and block recovery and show that the performance of a CNN degrades as we move closer to block recovery. Furthermore we will show how our proposed CSC based method is relatively unaffected by this difference in measurement matrix structure, even without retraining. Specifically we define a set of learning tasks parameterized by the scalar α. In each task the input is a measurement matrix $\mathbf{M}_i(\alpha)$ and the measured image $\mathbf{y}_i = \mathbf{M}_i(\alpha)\mathbf{z}_i$, and the goal is to recover the original image $\mathbf{z}_i$. We let $\mathbf{M}_i(\alpha) = \mathbf{M}_{inpainting}(1-\alpha) + \mathbf{M}_{block}\alpha$ where $\mathbf{M}_{inpainting}$ is a diagonal matrix which zeros out some pixels, and $\mathbf{M}_{block}$ is the previously described block diagonal transform. We use as data the Berkeley Segmentation Dataset (BSD-500) which provides a canonical set of high resolution images. Our baseline CNN is shown in figure 2.7, and to compare we use a three layer CSC model. We note that for the CNN we have to retrain for different values

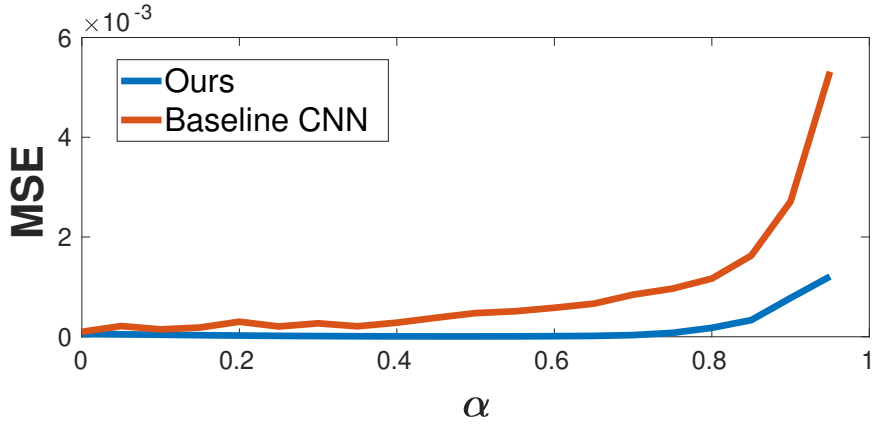


Figure 2.8: The results of the synthetic expperiment on BSD-500. For each value of α we train our CNN 5 times to convergence and then report the average of the top three validation results. In contrast we only train our model one time at $\alpha = 0.5$ which explains why the performance of our model peaks there since the hyper-parameters have been learned for that value.

of α but for the CSC model we simply train once at $\alpha = 0.5$. The results can be found in figure (2.8), validating our hypothesis. While interesting, this experiment wouldn't be useful if it didn't lead us to some practical application. The following experiments aim to do this by showing how two different computer vision tasks have this block diagonal structure.

2.5.2 JPEG Artifact Removal

In the previous section we showed how when the measurement matrix is block diagonal, we expect our CSC based method to perform better than a CNN. The synthetic experiment was contrived to demonstrate this, but we will now see how removing artifacts created by the original JPEG algorithm in fact has a similar form.

The compression algorithm operates on 8x8 blocks of the image independently. It is comprised of five steps: conversion to $\mathrm{YC}_b\mathrm{C}_r$ space, down-sampling of chroma channels, DCT-II transformation, rounding of DCT-II coefficients, and finally lossless compression of the remaining coefficients. The amount of rounding is controlled by a scalar parameter known as the quality factor, which varies from 1 to 100. The first three of these steps are linear transformations, and the final step is lossless and can therefore be discarded for our purposes. The fourth step creates the well known blocking artifacts, but is not linear. However as can be seen in Figure 2.6 the majority of the artifacts are created by the frequency components which are rounded to zero. Therefore we can make an approximation to the JPEG algorithm by ignoring the rounding on frequency components which are not zeroed. This approximation

is *locally linear* and can be expressed as $\mathbf{M}_{JPEG} = \mathbf{M}_{zeroing}\mathbf{M}_{DCT}\mathbf{M}_{downsample}$, where we have opted to perform the experiment in $\mathrm{YC}_b\mathrm{C}_r$ space.

Since the algorithm processes each block separately their collective action is very similar to the previous synthetic experiment. As such a CNN would be expected to perform very poorly if applied directly to the output of the algorithm, the DCT-II coefficients. To avoid this issue, other authors choose to manually undo the the DCT and downsampling operations, effectively applying the CNN to the decompressed image. This is the approach taken by Zheng *et al.* [52] which we will use as a comparison CNN architecture. We also compare against the recent work of Fu *et al.* [34], who also chose to apply convolutional sparse coding to this problem. However, there are several key differences between our approaches: First Fu *et al.* bases their architecture on *single layer* CSC. Second, Fu *et al.* applies LISTA to the objective, resulting in an network which is inspired by an optimization instead of explicitly performing it. Finally and most importantly, Fu *et al.* do not model the JPEG degradation and instead use the decompressed image as their input. As a result of these differences we find that our method uses far fewer parameters and achieves significantly better results. These findings are consistent with our hypothesis that explicit modeling of the measurement matrix is key when dealing with non-convolutional corruption.

We perform all experiments on the BSD-500 dataset for a range of quality factors. We also perform all experiments on the full $\mathrm{YC}_b\mathrm{C}_r$ image instead of on only the luminance channel. Since the method of Fu *et al.* [34] was designed for only a single channel input, we double the number of features in each convolution to compensate for the larger input.

In the first experiment we fix a quality factor and train each method to predict the original image from the compressed one. The results are show in table 2.2, which demonstrates that when the measurement matrix is closer to block diagonal our method achieves state of the art performance at quality factors 5 and 20 all while using much fewer parameters than the next closest model. Furthermore we are able to outperform both the deep sparse coding method and traditional multi-layer CSC by a wide margin at all quality factors. This is shown in the table as well as in the results of figure 2.11. That figure shows the results of a similar experiment except that we use identitcal model parameters to Sulam *et al.* , to demonstrate the effectiveness of learning the hyper-parameters directly from the data. Finally we performed an experiment where we mismatch the training and testing quality factors in order to measure the generalization of each model. Figure 2.10 shows the results of this experiment which demonstrate that due to our method specifically modeling the degradation, it outperforms all others by a wide margin when there is a mismatch between testing and training degradation.



Figure 2.9: Some example artifact removal results from BSD-500, zoomed in to highlight the detail. From left to right: Degraded input image, target image, our result

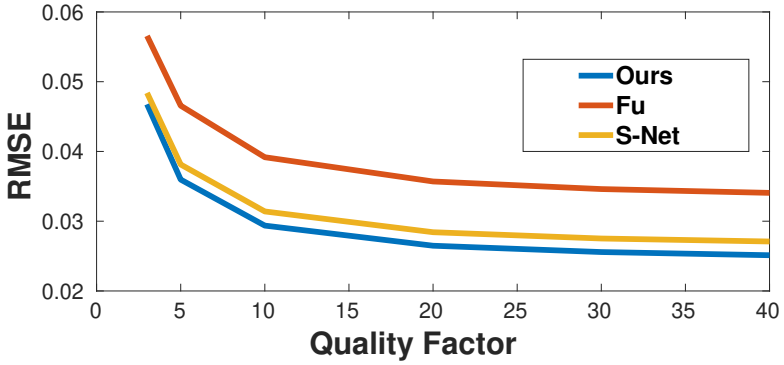


Figure 2.10: A comparison of our method and that of Zheng *et al.* on the JPEG artifact removal task. Note that as the degradation becomes more sever, that is the blocks of the measurement matrix become less diagonal, we see the same kind of fall off in performance as we did in the synthetic experiment.

2.5.3 Non-Rigid Trajectory Reconstruction

In non-rigid trajectory reconstruction, one seeks to recover the 3D trajectories of points from their 2D projections. Zhu *et al.* first proposed convolutional sparse coding for this problem in [53]. They demonstrate large performance gains through learning the filters directly from motion-capture data. However this requires manual tuning of the sparsity parameter which can be difficult to get right. In contrast, our method learns this parameter without tuning. Before presenting these results we will first give a brief description of the problem.

We refer the reader to Zhu *et al.* [53] for a description of how the trajetory reconstruction problem can be phrased as a sparse coding one, and present here just the equations for the input signal and measurement matrix.

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & & \\ & \ddots & \\ & & \mathbf{M}_F \end{bmatrix}, \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_F \end{bmatrix} \tag{2.15}$$

$$\tag{2.16}$$



Quality Factor	Ours	Sulam *et al.*
5	**0.039**	0.080
25	**0.069**	0.11

Figure 2.11: From top to bottom: image degraded at QF 5, our method's output, the method of Sulam *et al.* , the target image. The poor performance of the method of Sulam *et al.* can be explained by the fact that these methods are very sensitive to the choice of parameters. The dictionary is learned with one set of parameters but there is no clear way to modify them once we introduce the JPEG degradation. The table reports RMSE for our method and Sulam *et al.* at quality factors 5 and 25

where the $\mathbf{M}_i, \mathbf{y}_i$ encode the weak perspective cameras and 2D points. They demonstrate that learning the dictionary $\mathbf{D}$ on real motion capture data yields very impressive results which are still the state of the art on this task. Using the 3D ground truth points as supervision, we can train our end to end multi-layer CSC method for this task using the same formulation as described previously, except now our convolutions are in time instead of space.

	NRMSE	RMSE
Zhu *et al.*	0.025	35.91
14 Layer Trajectory CNN	0.0254	39.98
Ours 3 Layer	**0.0179**	**27.16**

Table 2.3: Results of multi-layer CSC (ML-CSC) on the trajectory reconstruction problem. We report Normalized RMSE in addition to RMSE, to better compare sequences of different scales. Please see [53] for a description of this metric.

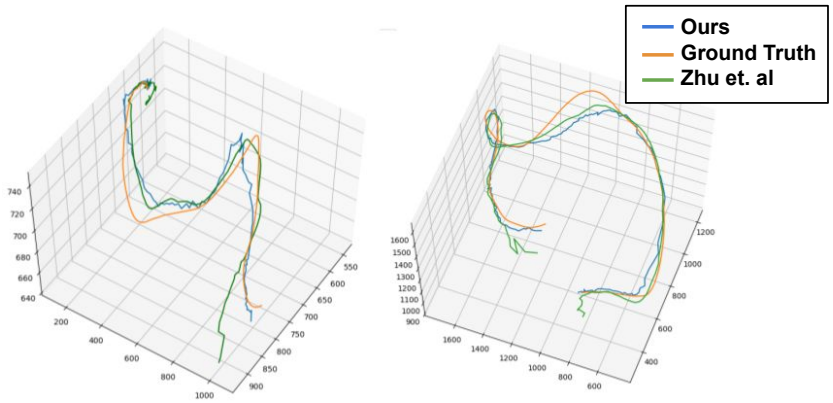


Figure 2.12: Some example trajectories and our reconstruction results

In our experiments we replicated the setup of Zhu *et al.* on the CMU motion capture dataset (mocap.cs.cmu.edu). We took each sequence, resampled them all to 30fps, created a synthetic orthographic camera which orbits the center of mass of the tracked points at a rate of π/s, and then cut the tracks into 150 frame sequences to form individual training examples. For validation, we hold out the sequences of subjects 14, 94, and 114, and use the rest for training. We can see from the results in Figure 2.3 that we are able to outperform those results by a wide margin.

We do not show results for a naive CNN on this task since they fail to produce reasonable predictions. This is in line with our hypothesis that inverse problems such as these are not amenable to simple CNN approaches. Instead we will show how we can derive an modification of the input from the multi layer CSC model that allows a simple CNN to achieve reasonable results on this task.

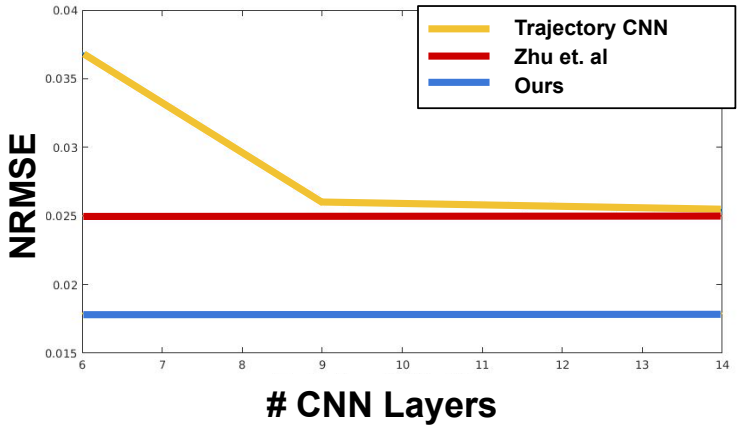


Figure 2.13: Performance vs Number of layers for our proposed Trajectory CNN



(a) Fu[34]: 0.0211|33.494|0.911 (b) S-Net: 0.0190|34.4|0.930 (c) Ours: 0.0150|36.5|0.946



(d) Ground Truth (e) JPEG

Figure 2.14: Qualitative results on BSD500. The captions contain RMSE, PSNR, and SSIM measures for each image

Recall that the first pass of our multi-layer CSC algorithm is equivalent to a feed forward CNN. Specifically we have that on the first iteration:

$$\mathbf{x}_i^{[0]} = \text{ReLU}(\frac{1}{L_i}\mathbf{D}_i^\top \mathbf{x}_{i-1}^{[0]} + \mathbf{b}_i). \tag{2.17}$$

However, we also have that when we have a measurement matrix $\mathbf{M}$ then the first dictionary $\mathbf{D}_1$ should be replaced by $\mathbf{M}\mathbf{D}_i$. This means that the first layer's first iteration is given by:

$$\mathbf{x}_1^{[0]} = \text{ReLU}(\frac{1}{L_i}\mathbf{D}_i^\top \mathbf{M}^\top \mathbf{y} + \mathbf{b}_1). \tag{2.18}$$

This leads to the observation that the input to the network is effectively $\mathbf{M}^\top \mathbf{y}$ instead of simply $\mathbf{y}$. We note that this is similar to how in the JPEG AR task, the input to the CNN was not the DCT coefficients but instead the decompressed image. Since the DCT transform is orthogonal, performing the inverse compression is very close to multiplying $\mathbf{y}$ by $\mathbf{M}^\top$. In general the product $\mathbf{M}^\top \mathbf{y}$ does not contain all of the information required to reconstruct $\mathbf{z}$, as we saw in the JPEG task as well. But, we will now see that in the non-rigid trajectory reconstruction task enough of the information is preserved that it allows a CNN to perform reasonably.

To give some intuition of why we expect this to work, let $\mathbf{p}_i$ be the original 3D point which is projected to $\mathbf{y}_i$ by orthogonal camera $\mathbf{M}_i$. Since the trajectories are smooth in time, the $\mathbf{p}_i$ have a convolutional structure appropriate for a CNN. This structure is destroyed when multiplied by $\mathbf{M}_i$. However, we note that $\mathbf{M}_i^T \mathbf{M}_i$ has the effect of back projecting the points to the 3D space, but with the depth set to 1. Since the cameras are smooth in time as well, the resulting trajectory is a distorted version of the original which is smooth. Thus we expect a CNN to perform well on this modified input. Our prediction is confirmed by the results in table 2.3. This network is able to achieve similar results to the CSC approach but requires many more layers to make up for the lack of iterations as shown in Figure 2.13.

Chapter 3

Un-Supervised Recovery Problems

3.1 Introduction

Research is often guided by improving benchmark performance, but we show that popular scene flow benchmarks may be guiding research in the wrong direction. Scene flow, the 3D analog of optical flow [107], can help autonomous vehicles identify moving objects. Most autonomous vehicles use LiDAR sensors for 3D perception, creating interest in estimating the dynamic motion between successive scans [88, 59, 57, 70, 94, 80]. If this task can be accomplished without relying on labeled data, it can give autonomous vehicles awareness of moving objects outside the detection taxonomy[96]. The potential for label-free detection has led to significant interest in self-supervised scene flow methods [113, 94, 57, 79]. We focus on this self-supervised setting and show that the standard benchmarks have several fundamental flaws. When evaluated more realistically, apparent benchmark improvements correspond to stalled or even decreasing real-world performance (Fig. 3.1).

Most work follows the evaluation framework from FlowNet3D [88], which primarily measures the average end-point-error (EPE) across FlyingThings3D [90] and KITTI-SF [92] (now typically referred to as stereoKITTI). These datasets have several issues: (1) stereoKITTI samples dynamic objects with an artificial pattern that differs from the pattern on static objects, inadvertently providing part of the "answer" to learning-based approaches. (2) Both FlyingThings3D and stereoKITTI ensure successive point clouds are in one-to-one correspondence, which is never the case for actual scans. (3) Both datasets contain an unrealistically high percentage of dynamic points. In contrast, real-world data is dominated by the background.

Together these issues obfuscate the main challenges of LiDAR scene flow: identifying the few non-static points and estimating their motions robustly given the lack of correspondences. We demonstrate the impact of these issues by evaluating a suite of top methods on several large-scale real-world datasets (Argoverse, NuScenes, Waymo), finding that performance is *negatively correlated* with performance on the standard benchmarks.

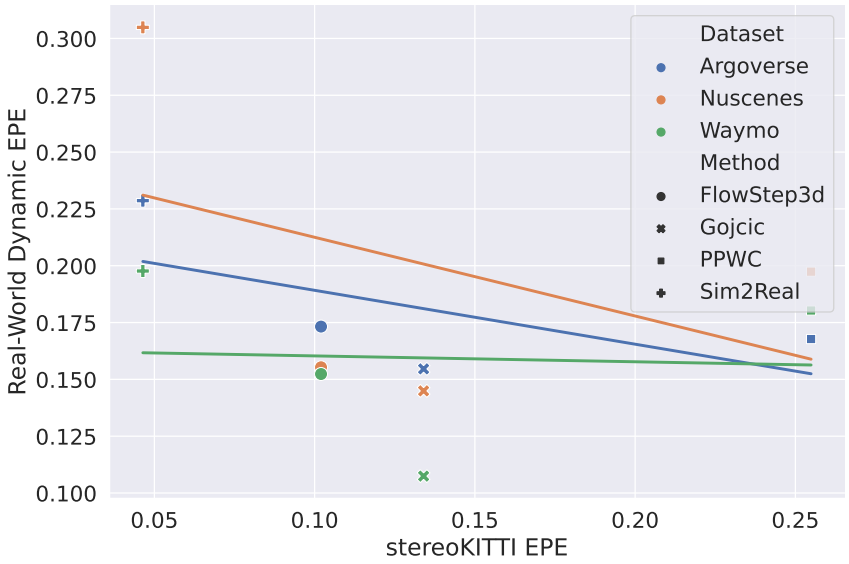


Figure 3.1: Recent self-supervised scene flow methods[113, 83, 79, 70] typically focus on performance on stereoKITTI[88]. We call attention to several problematic aspects of its semi-synthetic construction. When evaluated on real-world datasets (Waymo, Argoverse, NuScenes) we find a **negative correlation with stereoKITTI performance**. For each method and LiDAR dataset, we plot a point corresponding to the self-reported end-point error on stereoKITTI versus the end-point error on dynamic points of that method trained on real data. For each dataset, we plot the best-fit line to visualize the correlation.

We also believe these benchmarks have led researchers to ignore the virtues of classic optimization-based approaches. Many learning-based methods have been proposed for first estimating and removing ego-motion[106, 70], but we show that none outperform Iterative Closest Point (ICP). Furthermore, we show that the common pre- and post-processing steps of ground removal and enforcing piecewise-rigidity have a larger impact on performance than any learning strategy. We show this through a baseline method that combines these steps with a test-time optimization flow method. This baseline, without any learning, outperforms every method in our suite and outperforms all self-supervised methods on the self-reported NuScenes [90] and lidarKITTI [70] benchmarks. This leads to the conclusion that current learning methods fail to extract information not present in a single example despite using large amounts of data.

In summary, our main contributions are:

- An investigation of the weakness of current self-supervised scene flow evaluations.
- A *dataless* flow method which gives state-of-the-art results.

- A standard evaluation protocol and codebase along with a "model-zoo" of top methods as well as flow labels for Argoverse 2.0[1].

3.2 Related Work

Scene Flow: Scene flow was introduced by [108], who posed the problem in the stereo RGB setting and spawned a large body of subsequent work [56, 61, 72, 75, 73, 91, 101, 100, 110, 111]. A related problem is non-rigid registration [54, 63, 85, 77, 99], which is focused on fitting dense point clouds or meshes. We are interested in the setting without images, based purely on sparse LiDAR point clouds.

Optimization based LiDAR Scene Flow: Dataless LiDAR scene flow estimation was first proposed by Dewan *et al.* [64]. Inspired by a non-rigid registration method regularized by the graph Laplacian [66], Pontes *et al.* [102] created an improved method. Their results were further improved upon by Li *et al.* [87] with the implicit regularization of coordinate networks [55, 62, 93, 98, 105]. We use [87] as the backbone of our baseline and additionally employ coordinate networks for height-map estimation. Apart from these, the vast majority of recent methods have focused on deep learning based solutions.

Self and Weakly-Supervised LiDAR Scene Flow: "Just Go with the Flow" [94] demonstrated that using a combination of nearest-neighbor and cycle consistency losses was enough to train the FlowNet3D network, avoiding the reliance on labeled data. This led to many other works which adopted similar losses [113, 106, 83, 57]. Others made use of easier to acquire sources of supervision such as foreground/background segmentation masks [70, 65] or addressed the synthetic to real domain gap [79]. Of particular relevance are those methods which make use of ego-motion estimation and piecewise rigid representations [70, 65, 86]. We show that these steps are critical to good performance, but find that ICP vastly outperforms learned approaches and that piecewise rigidity is more effective as a post-processing step than as a loss regularizer.

Ground Segmentation: In robotics, ground segmentation has been studied as a subset of general dataless segmentation [95], traversable area identification [74], and as a pre-processing step for object detection, classification and tracking [114, 97, 78, 84]. Current scene flow methods do not make use of these sophisticated methods and instead rely on basic plane fitting [57, 71, 96]. In order to show how even small improvements to pre- and post- processing steps can drastically improve scene flow estimation, we propose a simple segmentation method based on coordinate networks.

[1] Code, weights, and outputs for all the evaluated methods will be released

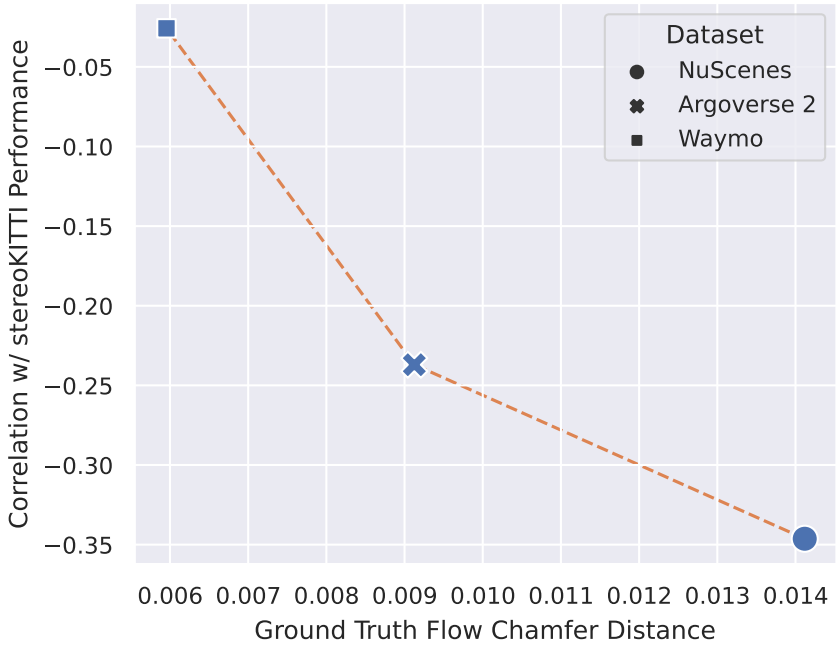


Figure 3.2: The correlation between performance on stereoKITTI and performance on real datasets as a function of how much those datasets violate the one-to-one correspondence assumption present in stereoKITTI. The more a dataset violates this assumption (higher chamfer distance due to sparser pointclouds), the worse the correlation with stereoKITTI performance.

3.3 Benchmark Issues

Most self-supervised flow estimation works inherit their evaluation protocol from [88], which is based on the synthetic FlyingThings3D [90] and stereoKITTI [92, 69]. Both were created for evaluating RGB-based flow methods and [88] extended them to point clouds by lifting the optical flow and depth annotations to 3D. These datasets and protocols suffer from three main deficiencies.

The one-to-one correspondence assumption removes the main challenge of working with LiDAR data. Since both FlyingThings3D and stereoKITTI are created by lifting optical flow annotations, the point clouds for each input pair are in one-to-one correspondence. For each 3D point $\mathbf{p}_i$ and ground truth 3D flow $\mathbf{f}_i$ in the first frame, there exists a 3D point $\mathbf{q}_i$ in the second frame such that $\mathbf{p}_i + \mathbf{f}_i = \mathbf{q}_i$. Real-world LiDAR scans do not have this property. The presence of correspondences fundamentally changes the LiDAR scene flow problem into a point-matching problem. It has been claimed that randomly sub-sampling the input breaks this correspondence. However, given the total number of points in each scan (90k) and the number of sub-sampled points (8192), there are still an expected 745 corresponding points in

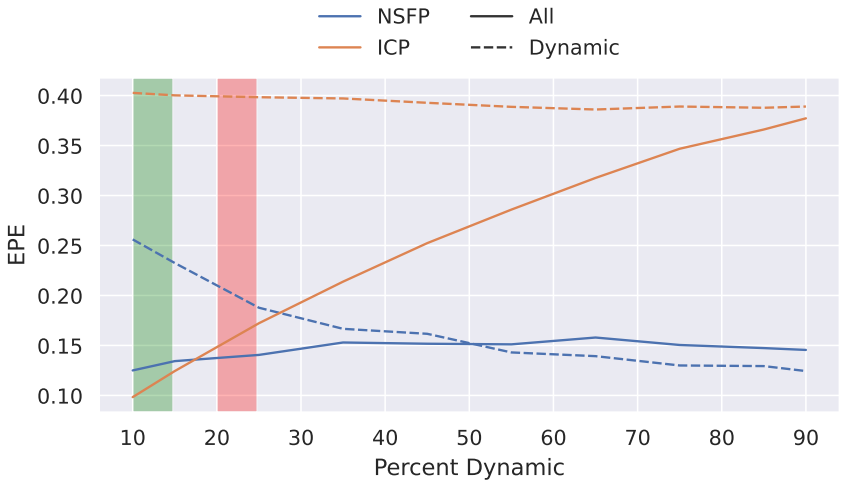


Figure 3.3: The performance of ICP and NSFP versus the ratio of dynamic points in each example. The green region indicates the ratio found in real data and the red region indicates the ratio found in the stereoKITTI dataset. The unrealistic dynamic ratio of stereoKITTI causes ICP to appear to perform worse than it does on real-world data such as Argoverse.

the input. Since each example has only a handful of independent motions, these correspondences are enough to constrain the solution.

This problem is especially severe for self-supervised scene flow methods trained using the chamfer distance, a symmetric extension of the nearest neighbor distance[94]. Essentially, each method deforms the first point cloud using the predicted flows and then computes the distance to the second point cloud. The chamfer distance is an excellent self-supervised objective for point clouds with one-to-one correspondences since the ground truth flow achieves the minimum distance of 0. However, for real-world point clouds with no correspondences the chamfer distance becomes a weaker proxy, and even the ground truth flow will have a non-zero distance. We can quantify this effect by computing the chamfer distance of the ground truth flows for several real-world datasets. In Fig. 3.2 we show the relationship between that quantity and the slopes of the best-fit lines shown in Fig. 3.1. As can be seen, the more a dataset violates the one-to-one assumption, the more *good* performance on stereoKITTI indicates *poor* performance on that dataset.

Current benchmarks' unrealistic rates of dynamic motion make estimating ego-motion seem harder than it is. In any scene, some portion of the measured points belong to the static background rather than dynamically moving objects. In real-world LiDAR scans such as those in NuScenes [60], Waymo [80], or Argoverse 2 [112], the percentage of static points is very high, ranging from 85-100% (Fig. 3.4). In contrast, FlyingThings3D has 0% static points since it consists of flying things, and

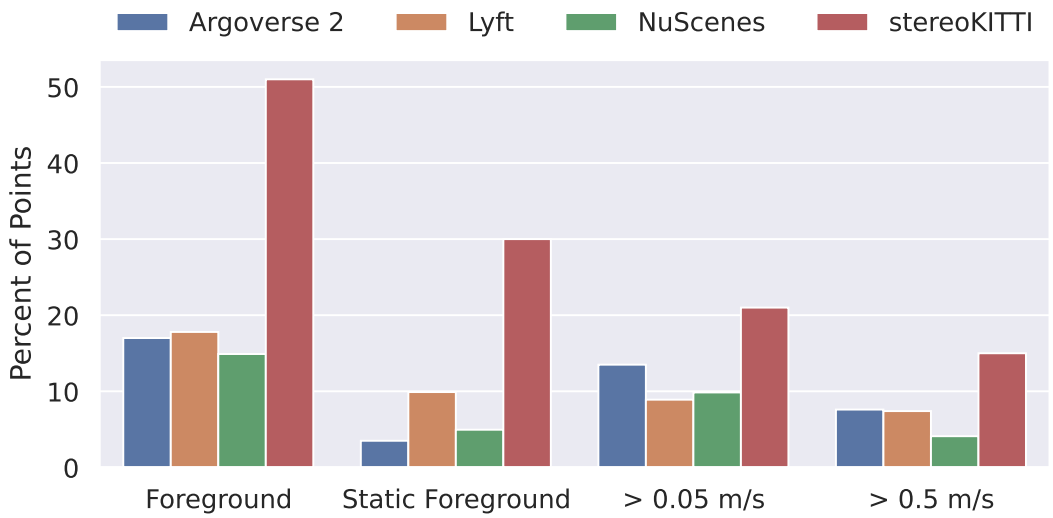


Figure 3.4: An analysis of the motion profile of points found in various autonomous driving datasets. The leftmost column is the percentage of points belonging to any tracked object. The right three columns show those points separated by their speed once ego-motion has been removed. All three datasets have a very low rate of dynamic foreground points, in contrast to the typical stereoKITTI benchmark.

the KITTI-SF dataset has approximately 75-85% static points. Dealing with this data imbalance is one of the key difficulties of self-supervised flow estimation but it is not present in the popular benchmarks. This discrepancy also explains why ICP has been ignored as a technique for ego-motion estimation[88].

In order to understand why recent methods have neglected ICP, we need to understand how the amount of dynamic points in a scene affects the performance of ICP. We manually re-sampled each example in the Argoverse 2.0 validation dataset to take a specified number of dynamic and static points and then ran a test-time optimisation scene flow method [87] and ICP [61]. The results (Fig. 3.3) show that when looking at the total EPE averaged over the entire dataset, like most evaluations, the performance of ICP steadily degrades as the percentage of dynamic points increases. At 20% dynamic points, double the rate of real-world data but approximately the same ratio as KITTI-SF, ICP's performance has degraded to significantly below the performance of NSFP. This explains why early methods which did include ICP in their comparisons [88] were able to outperform it, leading future methods to leave it out even as more realistic data became available.

The sampling pattern of dynamic objects in stereoKITTI gives away the answer in the input. The ground truth flow and disparity of KITTI-SF were created by fitting 3D models to the LiDAR points and 2D annotations [91]. This gives the dynamic objects a distinctive dense sampling pattern that separates them from the

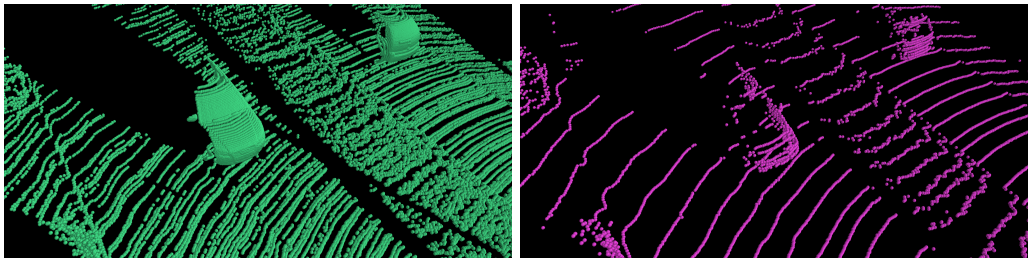


Figure 3.5: Comparison of sampling patterns from KITTI-SF (**left**) versus the corresponding real LiDAR scan (**right**). KITTI-SF uses dense CAD models for foreground objects, which makes it easier for learning-based methods to find them among the sparse LiDAR background points.

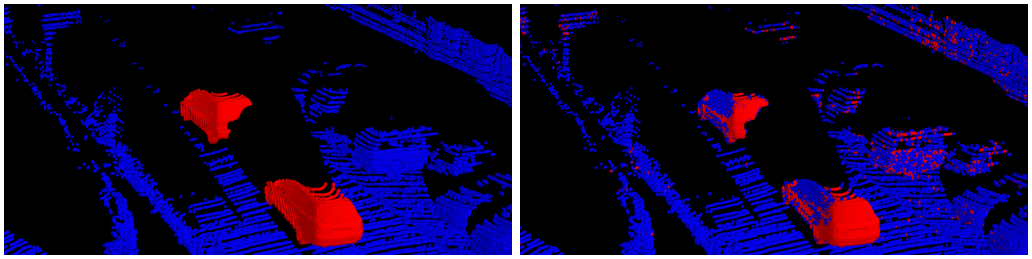


Figure 3.6: An example ground truth segmentation from KITTI-SF (**left**) compared with (**right**) an example prediction from our local model. Note how the model can even identify parked vs moving cars since only moving vehicles have the dense sampling pattern.

background (see Fig. 3.5). However, determining which points belong to moving objects is one of the main challenges and the sampling pattern effectively gives away the answer in the input. We demonstrate that learning-based system can simply identify moving objects by the sampling pattern without learning anything about the motion.

Model: We use a model that takes a small amount of local context and *no information from the second frame*, to predict motion segmentation for a point $\mathbf{p}$. That is, the input to our network is the set of vectors $\{\mathbf{q} - \mathbf{p} \mid \forall \mathbf{q} \in \mathcal{N}_r(\mathbf{p})\}$ where $\mathcal{N}_r$ is a ball of radius r centered at $\mathbf{p}$. We use $r = 0.05\,\text{m}$ in our experiment. Then we use a standard PointNet architecture to predict a binary label for the point.

Training: We train our model on the first 150 examples of KITTI-SF and test on the remaining 50. Note this is essentially the split used by FlowNet3D[88] and the recent RigidFlow[86] for fine-tuning. We train our network using a cross-entropy loss weighted by the inverse frequency of each label. For comparison, we perform the same experiment on real LiDAR scans from Argoverse. We sample 150 scans from the train set and 50 scans from the validation set.

Results: Using only information in a 5 cm ball around each point we are able to

segment KITTI-SF with high accuracy. We achieve a mean intersection over union on the foreground of 0.83 and 0.97 on the foreground. An example is visualized in Fig. 3.6. In comparison when we attempt this on real LiDAR scans that do not have a biased sampling pattern, we achieve foreground and background mean intersection over union scores of 0.16 and 0.81 respectively. This demonstrates that the sampling pattern of KITTI-SF makes it trivial to identify foreground points and should not be used for training or validation.

3.3.1 Real-World Flow Evaluation

Due to these deficiencies, recent works have made progress on improving results on these benchmarks without improving results on real-world data (Fig. 3.1). Some of the problems could perhaps be addressed through re-sampling, but doing so requires modifying the data based on the ground truth static-vs-dynamic labels. This runs counter to the goal of creating methods that operate on raw, unlabeled data.

Some works have evaluated on real LiDAR data by transferring synthetic labels [70], or more commonly by computing flows from object-level tracks [57, 80, 79, 87]. However, those evaluations have been presented as auxiliary results [57, 87] with limited comparisons to existing methods, or without comparison entirely [80]. As a result, KITTI-SF and FlyingThing3D remain the standard benchmarks. We argue that evaluating on real-data should be the "gold-standard" for scene flow as opposed to synthetic benchmarks.

To further emphasise how top learning-based methods fail when evaluated properly on real data, we also demonstrate that they can be outperformed by a simple test-time optimization baseline. We construct this baseline by making small improvements to standard pre- and post-processing steps and wrapping them around neural scene flow prior.

3.4 Baseline

Our pipeline (Fig. 4.4), consists mainly of pre- and post-processing steps around the test-time flow optimization method of [87].

Motion Compensation: Since real-world scenes consist mostly of static background objects, removing the ego-motion of the sensor makes estimating the dynamic motion significantly easier (Fig. 3.8). For datasets such as Argoverse or Waymo, the ego-motion is provided and we transform the first scan into the coordinate frame of the second. NuScenes and lidarKITTI also have this information but previous work has opted to not use it. In this case, we use ICP [61, 109] to first estimate the ego-motion. We demonstrate experimentally that this is much more effective than learned approaches.

Ground Removal: The sampling pattern of a LiDAR sensor creates "swimming" artifacts as the sensor moves. These artifacts create the appearance of motion and

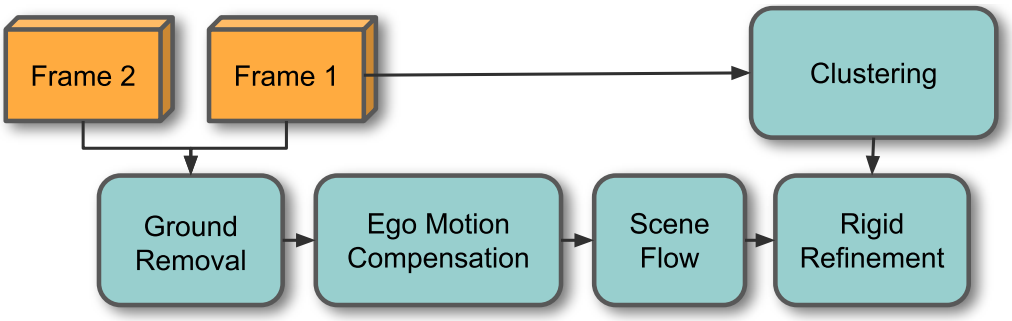


Figure 3.7: Our proposed pipeline for LiDAR scene flow based largely on classical pre- and post-processing.

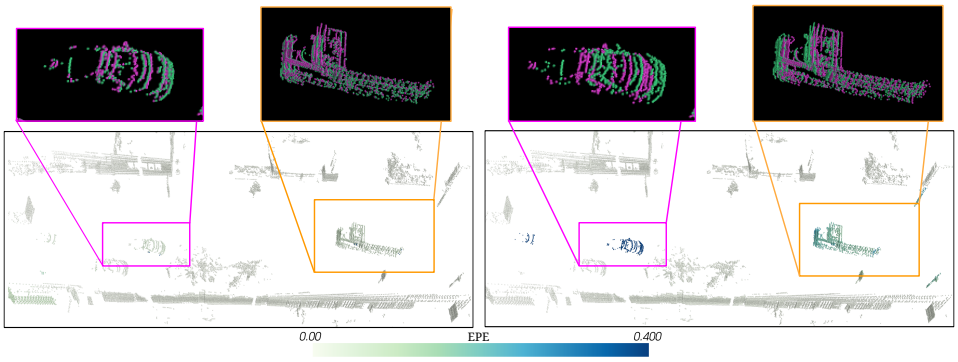


Figure 3.8: Comparison between NSFP with our proposed pre and post processing steps (**left**) and standard NSFP (**right**). In the (**bottom**) views points are color coded by EPE. The (**top**) detail views show the first and second frames aligned by the predicted flow. NSFP struggles to represent both foreground and background motion. We find that first using ICP to remove ego-motion greatly improves the estimates on dynamic points.

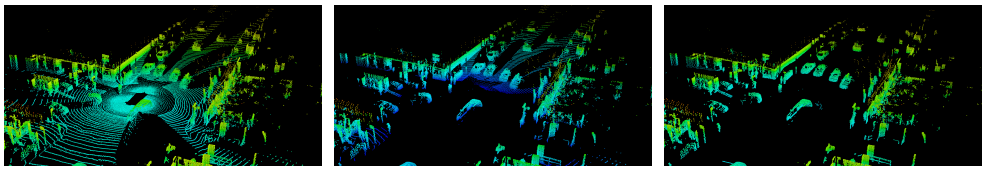


Figure 3.9: An example of our ground removal technique on a Waymo [80] scene with a non-planar ground. In all images, color coding indicates the height of each point. From left to right: the input point cloud, the result of thresholding, our result.

present a large problem for the nearest neighbor loss function used by almost all self-supervised methods. The largest artifacts come from the ground, leading most methods to remove them by height thresholding or by fitting a plane [57, 88]. This can fail when the ground changes elevation either from hills or even sidewalks, see Fig. 3.9. We propose an improvement based on analyzing the assumptions behind the current approaches. These assumptions are:

1. The sensor has been calibrated such that the ground can be represented as a height map $h = f(x, y)$.
2. Except for a small number of noise returns, all measured points (x, y, z) satisfy $z \geq f(x, y)$
3. The ground function $f(x, y)$ is in some way "simple". Existing methods make use of this assumption by requiring $f(x, y) = c$ for thresholding or $f(x, y) = ax + by + c$ for plane fitting.

The issue comes from an overly strict application of assumption (3). Instead, we allow our height map to be *piecewise linear* rather than linear. To represent our piecewise linear height map we use a 3-layer coordinate network with ReLU activations and 64 hidden units per layer. To fit it we use assumption (2) to design a one-sided robust loss:

$$\mathcal{L}_{height}(h, z) = \begin{cases} (h - z)^2 & z < h \\ \text{Huber}(h, z) & h \leq z \end{cases}. \tag{3.1}$$

The Huber function [76] allows the loss to ignore points high above the ground. For each point cloud the parameters of our height-map θ_h are found by optimizing

$$\min_{\theta_h} \sum_{i=1}^{N} \mathcal{L}_{height}(f_{\theta_h}(x_i, y_i), z_i). \tag{3.2}$$

Finally, any points which are less than 0.3m above our predicted ground are removed.

Scene Flow Estimation: Once the ego-motion and ground points have been removed, we estimate the scene flow using the method of Li *et al.* [87]. Briefly, this means that we represent the forwards and backward flow using two coordinate networks $f_{\theta_{fw}}, f_{\theta_{bw}} \in \mathbb{R}^{N\times 3} \to \mathbb{R}^{N\times 3}$ which are optimized with gradient descent on the objective:

$$\min_{\theta_{fw}, \theta_{bw}} \mathcal{C}(g_{\theta_{fw}}(\mathbf{P}^t), \mathbf{P}^{t+\Delta}) + \mathcal{C}(g_{\theta_{bw}}(g_{\theta_{fw}}(\mathbf{P}^t)), \mathbf{P}^t). \tag{3.3}$$

For compactness, we have let $g(\mathbf{X}) = f(\mathbf{X}) + \mathbf{X}$ and $\mathcal{C}$ be the truncated symmetric chamfer distance as described in [87]. We differ from the original method in that we do not use ℓ_2 regularization on the weights. We find that when combined with motion compensation that regularization leads to zero flow predictions everywhere.

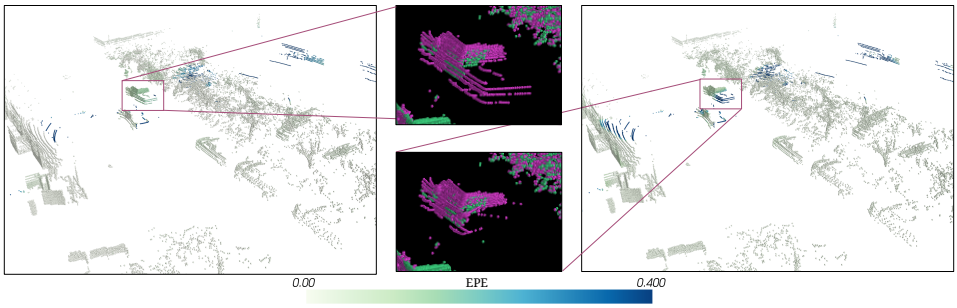


Figure 3.10: Optimizing for the standard nearest neighbor self-supervised loss can cause mis-predictions in the presence of strong occlusions. Here, the bed of the truck is collapsed into the cab by NSFP (**left**). Our RANSAC non-rigid refinement step can fix this type of error (**right**).

Rigid Refinement: Piecewise rigidity has been a common choice of prior for scene flow estimation and has been widely used for LiDAR scene flow [70, 65, 57]. Many learning-based methods require differentiable rigid refinement for loss functions, but we show that simply fitting rigid motion to the final predictions performs better. This is similar to the method of [70] but we use RANSAC to filter outliers. First we use DBSCAN [67] to produce a set of clusters $\{\mathcal{V}_j = \{\mathbf{p}_k\}_{k=1}^{K}\}_{j=1}^{J}$. Then for each cluster, we use RANSAC [68] to fit a rigid model to flow predictions for that cluster. That is for each RANSAC iteration we randomly sample 3 points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ (the minimum required to get a unique solution) and their associated flow vectors $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$ and then fit rigid motion parameters by solving:

$$\min_{\mathbf{R}\in SO(3),\mathbf{t}\in\mathbb{R}^3} \sum_{i=1}^{3} \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - (\mathbf{p}_i + \mathbf{f}_i)\|_2^2, \tag{3.4}$$

with the Kabsch algorithm [81]. We then compute the norm of the difference between the raw and rigid flows, considering any below a threshold to be inliers. At the end of T iterations the rigid parameters with the highest number of inliers are selected and the parameters are recomputed with respect to the inlier set producing $\mathbf{R}^*, \mathbf{t}^*$. Since we know that with motion-compensated inputs the vast majority of flows should be zero, we further refine the flows by setting the rigid motion parameters to the identity transform if $\|\mathbf{t}^*\|_2$ is below a threshold. Then all points in the cluster are assigned the flow $\mathbf{f}_i = \mathbf{R}^*\mathbf{p}_i + \mathbf{t}^* - \mathbf{p}_i$. Points that were not assigned to any cluster by DBSCAN have their predictions unchanged. The effect of this step can be seen in Fig. 3.10.

	Supervision	EPE				AccR	AccS
		Avg	Dynamic	Static		Dynamic	Dynamic
			FG	FG	BG	FG	FG
Gojcic [70]	Weak	0.083	0.155	0.064	0.032	0.650	0.368
EgoFlow [106]	Weak	0.205	0.447	0.079	0.090	0.111	0.018
Sim2Real [79]	Synth	0.157	0.229	0.106	0.137	0.565	0.254
PPWC [113]	Self	0.130	0.168	0.092	0.129	0.556	0.229
FlowStep3D [83]	Self	0.161	0.173	0.132	0.176	0.553	0.248
Odometry	None	0.198	0.583	0.010	0.000	0.108	0.002
ICP [61]	None	0.204	0.557	**0.025**	**0.028**	0.112	0.015
NSFP [87]	None	0.088	0.193	0.033	0.039	0.542	0.327
Ours	None	**0.055**	**0.105**	0.033	**0.028**	**0.777**	**0.537**

Table 3.1: Quantitative results on Argoverse 2. Our baseline predicts the motion of dynamic objects by 30% over [70], despite that method using ground truth foreground masks for training. We also see from the static background EPE that ICP outperforms learning-based methods at predicting ego-motion.

	Supervision	EPE				AccR	AccS
		Avg	Dynamic	Static		Dynamic	Dynamic
			FG	FG	BG	FG	FG
Gojcic [70]	Weak	0.084	0.145	0.060	0.047	0.714	0.436
EgoFlow [106]	Weak	0.236	0.520	0.074	0.114	0.105	0.022
Sim2Real [79]	Synth	0.214	0.305	0.143	0.194	0.426	0.155
PPWC [113]	Self	0.156	0.197	0.097	0.173	0.468	0.183
FlowStep3D [83]	Self	0.156	0.155	0.090	0.224	0.660	0.370
ICP	None	0.176	0.450	0.033	**0.046**	0.151	0.047
NSFP	None	0.088	0.130	0.034	0.101	0.711	0.447
Ours	None	**0.055**	**0.061**	**0.020**	0.083	**0.891**	**0.681**

Table 3.2: Quantitative results on NuScenes. Our baseline halves the EPE on dynamic objects when compared to the next best method (NSFP). Since our baseline also uses NSFP as its backbone, this indicates that pre- and post-processing steps can have an enormous impact. Again we also see that ICP has the best ego-motion prediction.

3.5 Evaluation

Our central claim is that the use of popular benchmarks causes leading methods to degrade in quality when evaluated on real-world data. The main result in support of this is shown in Fig. 3.1; here we detail the procedure used to select, train, and evaluate the tested methods (Sec. 3.5.1). We also discuss the performance of our baseline, which we find outperforms all the tested methods, as well as validate these results through comparison to other authors's self-reported metrics on NuScenes and lidarKITTI (Sec. 3.5.2). Finally, we evaluate our ground segmentation method

	Supervision	EPE				AccR	AccS
		Avg	Dynamic	Static		Dynamic	Dynamic
			FG	FG	BG	FG	FG
Gojcic [70]	Weak	0.059	0.108	0.045	**0.025**	0.844	0.584
EgoFlow [106]	Weak	0.183	0.390	0.069	0.089	0.178	0.046
Sim2Real [79]	Synth	0.132	0.180	0.075	0.142	0.497	0.217
PPWC [113]	Self	0.132	0.180	0.075	0.142	0.497	0.217
FlowStep3D [83]	Self	0.169	0.152	0.123	0.232	0.708	0.405
ICP	None	0.192	0.498	0.022	0.055	0.172	0.047
NSFP	None	0.100	0.171	0.021	0.108	0.539	0.331
Ours	None	**0.041**	**0.073**	**0.013**	0.039	**0.877**	**0.726**

Table 3.3: Quantitative results on Waymo. Our baseline outperforms all tested methods but notably in this instance [70] performs better than ICP at predicting the background ego-motion.

	Moving		Static	50-50
	EPE	Accuracy Relax	EPE	EPE
Zero	0.6381	0.1632	0.5248	0.5814
ICP	0.2101	0.6151	**0.0290**	0.1196
PPWC	0.3539	0.2543	0.1974	0.2756
EgoFlow	0.7399	0.0000	0.0570	0.3985
SLIM (U)	0.1050	0.7365	0.0925	0.0987
Ours	**0.0625**	**0.894**	0.0660	0.064
SLIM (S)	0.0702	0.8921	0.0499	**0.0600**

Table 3.4: The results of our baseline on NuScenes as evaluated by Baur *et al.* [57]. U and S refer to the unsupervised and fully-supervised versions of their method. Our dataless baseline outperforms all self-supervised methods and even achieves comparable and superior results to the fully supervised network.

	Supervision	EPE	AccS	AccR
PPWC [113]	Full	0.710	0.114	0.219
FLOT [103]	Full	0.773	0.084	0.177
MeteorNet [89]	Full	0.277	/	/
Gojcic [70]	Weak	0.133	0.460	0.746
Gojcic++	Weak	0.102	0.686	0.819
Dong (Waymo Open) [65]	Weak	0.077	0.812	0.906
Dong (Semantic KITTI)	Weak	0.065	0.857	0.940
Ours	None	**0.061**	**0.917**	**0.962**

Table 3.5: Results of our method on lidarKITTI w/ ground [70]. We outperform all existing methods without using any training data.

(Sec. 3.5.3).

3.5.1 Training and Evaluation Protocol

Evaluation Metrics: We use a set of standard metrics:

- **EPE:** Average end-point-error *i.e.* the ℓ_2 norm of the difference between predicted and ground truth flow.
- **Accuracy Relax:** Ratio of predictions with absolute EPE less than 0.1m or relative error below 0.1.
- **Accuracy Strict:** Same as Accuracy Relax but with a threshold of 0.05.

Rather than averaging these metrics over the entire dataset, we break them into three classes of points: dynamic foreground, static foreground, and static background. Points are considered belonging to the foreground if they are contained in the bounding box of some tracked object and they are considered dynamic if they have a flow magnitude of at least $0.5\,\mathrm{m\,s^{-1}}$. The choice of threshold is discussed in the supplement. Separating the metrics in this way is vital due to the low ratio of dynamic to static points, without it the metrics are dominated by performance on the static background. In order to produce a single number for ranking purposes we combine the EPE results on the classes into a three-way average similar to [57]. In order to save computation we evaluate on a subset on of each dataset's validation split. We use the entire NuScenes validation set, and every 5th frame of the Argoverse and Waymo validation sets.

Method Selection: We chose to evaluate 6 methods to serve as a thorough exploration of recent research. Methods were chosen if they presented a self or weakly supervised method, appeared in a recent conference, and made available a working implementation of their method. This led us to choose: PointPWC Net [113], EgoFlow [106], FlowStep3D [83], NSFP [87], Sim2Real [79], and Gojcic *et al.* [70]. Additionally, we include as baselines an off-the-shelf ICP [58] implementation [109]. Most of the baseline methods clip points to a depth of 35m [113, 79, 106, 83]. To make comparisons fair we only include points contained in a 70m square around the sensor as was done in [57].

Training Procedure: As much as possible we attempted to use the same training strategy as the original authors, but we also wished to enforce standardization on the amount of computing resources allocated to each method. As a result, we chose to adopt the two-stage training regime from [113]. First, we train for seven days on a quarter of the whole training set followed by five days of fine-tuning on the entire dataset. Each method was trained using the largest batch size possible on a single NVIDIA T4 GPU (some methods were not set up for multi-GPU training), and using the authors' optimizer and learning rate schedule. Methods that were able to include ground truth ego-motion in their loss formulation were also given that

information. We also used ground truth foreground/background masks to train the weakly supervised method [70]. The only method that required substantial changes was [79] as its training is based on transferring information from a synthetic labeled dataset. The synthetic data comes from a virtual depth camera, so we clip the LiDAR points to match the field of view.

Results: Our results can be found in tables 3.1, 3.2, and 3.3. The main result in Fig. 3.1 was created by taking the dynamic foreground error for each learning method/dataset combination and plotting it against their self-reported stereoKITTI error. We use the dynamic foreground EPE since this is the hardest and most important component of the real-world scene flow problem. We remove EgoFlow from these plots since it is a large outlier in terms of both real-world dynamic EPE and stereoKITTI EPE.

We also claim that synthetic benchmarks are causing researchers to focus on the wrong problems, which we test by comparing the tested methods to our simple test-time optimization baseline. Validating this claim, we can observe that our baseline outperforms all tested methods despite not using any training data; even Gojcic *et al.* who use ground truth foreground masks. Each tested method claims some learning-based novelty as its main contribution, which is then shown to be effective through experiments on KITTI-SF and FlyingThings3D. However, none can outperform a carefully designed baseline when evaluated on real data. Further validating this claim is the fact that no method was able to match the performance of ICP at predicting the ego-motion. This is in spite of the fact that several of the methods [70, 57, 106] explicitly claim ego-motion predictions as a benefit of their architectures. As discussed in Sec. 3.3, this results from the unrealistic dynamic ratio found in the standard benchmarks. Gojcic *et al.* comes the closest to outperforming ICP and does so on Waymo. However, this is because it essentially incorporates ICP as a part of its final test-time refinement of the ego-motion.

3.5.2 Existing Benchmark Comparison

There may be some concern that the tested methods performed poorly compared to the baseline due to a lack of tuning of hyperparameters. To address this concern, we also compare our baseline to self-reported metrics on two real-world datsets: NuScenes as evaluted in [57] and lidarKITTI which was introduced by the Gojcic *et al.* [70].

NuScenes: The authors of SLIM [57] also reported numbers on real-world data. As can be seen in Tab. 3.4 our method vastly outperforms SLIM and the other methods they tested on the main 50-50 EPE metric, dynamic EPE, and dynamic accuracy. Again, ICP performs far better than every other method on static points. We also include the SLIM results when trained in a supervised manner and show that we achieve comparable and some superior results, despite not using any training data.

lidarKITTI: lidarKITTI [70] was generated by associating KITTI-SF labels with

the raw LiDAR scans. We use the version with ground points as ground removal is a component of our method. Ground point flow is set to the ego-motion. The results are shown in Tab. 3.5 and further confirm that our baseline performs well even compared to self-reported numbers. It substantially outperforms all but one of the existing self- and weakly-supervised methods.

3.5.3 Ground Segmentation

We close with an evaluation of our ground segmentation method. First, we present a qualitative example (Fig. 3.9) showing our method effectively handling a scene with a non-planar ground. To quantitatively analyze our method we need to consider what constitutes failure. The subtlety lies in the ill-defined nature of the ground, making it impossible to define normal precision and recall metrics. Given that our goal is motion estimation, the failure we are most concerned with is classifying a dynamic point as belonging to the ground. Therefore we look at the rate at which we make this error. On NuScenes we find that 99.3% of the time points which are classified as ground are in fact static and achieve a rate of 99.4% on Waymo.

3.6 Conclusion

We re-examined the evaluations of self- and weakly- supervised scene flow methods in the context of autonomous driving and found several deficiencies. We claimed that these deficiencies are impacting the quality and types of recently proposed methods. To provide evidence for this claim we evaluated a large number of top methods on several real-world datasets. We found a negative correlation between performance on the standard stereoKITTI benchmark and performance on all of the real-world datasets. Additionally, we proposed a dataless estimation technique that far outperformed the existing approaches, demonstrating that focus on the current benchmarks is causing researchers to ignore effective methods. Given that our baseline is based on pre- and post-processing techniques, we believe that other methods not based on test-time optimization will also benefit from them.

3.7 Student/Teacher Distillation

We propose *Scene Flow via Distillation* (SFvD), a simple, scalable distillation framework that creates a new class of scene flow estimators by using a label-free optimization method to produce pseudo-labels to supervise a feedforward model (Figure 3.11). While conceptually simple, efficiently instantiating SFvD requires careful construction; most online optimization methods and feedforward architectures are unable to efficiently scale to full-size point clouds (Section 3.7.1).

Based on our scalability analysis, we propose *Zero-Label Scalable Scene Flow* (ZeroFlow), a family of scene flow models based on SFvD that produces fast, **state-**

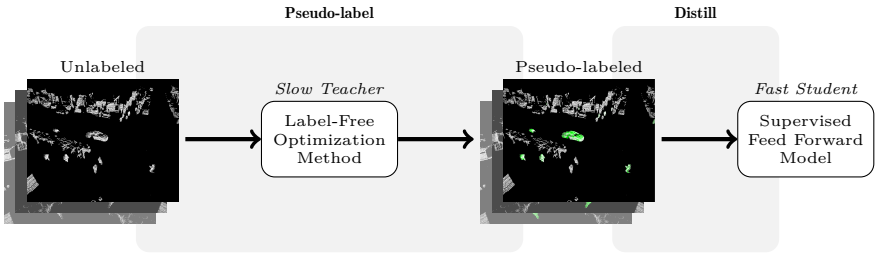


Figure 3.11: The *Scene Flow via Distillation* (SFvD) framework, which describes a new class of scene flow methods that produce high quality, human label-free flow at the speed of feedforward networks.

of-the-art scene flow estimates for full-size point clouds without any human labels. ZeroFlow uses Neural Scene Flow prior (NSFP) [127] to generate high quality, label-free pseudo-labels on full-size point clouds (Section 3.7.2) and FastFlow3D [122] for efficient inference (Section 3.7.3).

3.7.1 Scaling Scene Flow via Distillation to Large Point Clouds

Popular AV datasets including Argoverse 2 ([144], collected with dual Velodyne VLP-32 sensors) and Waymo Open ([138], collected with a proprietary lidar sensor and subsampled) have full-size point clouds with an average of 52,000 and 79,000 points per frame, respectively, after ground plane removal. For practical applications, sensors such as the Velodyne VLP-128 in dual return mode produce up to 480,000 points per sweep [140] and proprietary sensors at full resolution can produce well over 1 million points per sweep. Thus, scene flow methods must be able to process many points in real-world applications.

Unfortunately, most existing methods focus strictly on scene flow *quality* for toy-sized point clouds, constructed by randomly subsampling full point clouds down to 8,192 points [79, 106, 113, 83, 128, 127]. As we are motivated by real-world applications, we instead target scene flow estimation for the full-sized point cloud, making architectural efficiency of paramount importance. As an example of stark differences between feedforward architectures, FastFlow3D [122], which uses a PointPillar-style encoder [124], can process 1 million points in under 100 ms on an NVIDIA Tesla P1000 GPU (making it real-time for a 10Hz LiDAR), while methods like FlowNet3D [128] take almost 4 seconds to process the same point cloud.

We design our approach to efficiently process full-size point clouds. For SFvD's pseudo-labeling step, speed is less of a concern; pseudo-labeling each point cloud pair is offline and highly parallelizable. High-quality methods like Neural Scene Flow Prior (NSFP, [127]) require only a modest amount of GPU memory (under 3GB) when estimating scene flow on point clouds with 70K points, enabling fast and low-cost pseudo-labeling using a cluster of commodity GPUs; as an example,

pseudo-labeling the Argoverse 2 train split with NSFP is over 1000× cheaper than human annotation. The efficiency of SFvD's student feedforward model *is* critical, as it determines both the method's test-time speed and its training speed (faster training enables scaling to larger datasets), motivating models that can efficiently process full-size point clouds.

3.7.2 Neural Scene Flow Prior is a Slow Teacher

Neural Scene Flow Prior (NSFP, [127]) is an optimization-based approach to scene flow estimation. Notably, it does not use ground truth labels to generate high quality flows, instead relying upon strong priors in its learnable function class (determined by the coordinate network's architecture) and optimization objective (Equation 3.5). Point residuals are fit per point cloud pair P_t, P_{t+1} at test-time by randomly initializing two MLPs; one to describe the forward flow $\hat{F}^+$ from P_t to P_{t+1}, and one to describe the reverse flow $\hat{F}^-$ from $P_t + \hat{F}_{t,t+1}$ to P_t in order to impose cycle consistency. The forward flow $\hat{F}^+$ and backward flow $\hat{F}^-$ are optimized jointly to minimize

$$\text{TruncatedChamfer}(P_t + \hat{F}^+, P_{t+1}) + \text{TruncatedChamfer}(P_t + \hat{F}^+ + \hat{F}^-, P_t) \ , \tag{3.5}$$

where TruncatedChamfer is the standard Chamfer distance with per-point distances above 2 meters set to zero to reduce the influence of outliers.

NSFP is able to produce high-quality scene flow estimations due to its choice of coordinate network architecture and use of cycle consistency constraint. The coordinate network's learnable function class is expressive enough to fit the low frequency signal of residuals for moving objects while restrictive enough to avoid fitting the high frequency noise from TruncatedChamfer, and the cycle consistency constraint acts as a local smoothness regularizer for the forward flow, as any shattering effects in the forward flow are penalized by the backwards flow. NSFP provides high quality estimates on full-size point clouds, so we select NSFP for ZeroFlow's pseudo-label step of SFvD.

3.7.3 FastFlow3D is a Fast Student

FastFlow3D [122] is an efficient feedforward method that learns using human supervisory labels $F^*_{t,t+1}$ and per-point foreground / background class labels. FastFlow3D's loss minimizes a variation of the End-Point Error that reduces the importance of annotated background points, thus minimizing

$$\frac{1}{\|P_t\|} \sum_{p \in P_t} \sigma(p) \|\hat{F}_{t,t+1}(p) - F^*_{t,t+1}(p)\|_2 \tag{3.6}$$

where

$$\sigma(p) = \begin{cases} 1 & \text{if } p \in \text{Foreground} \\ 0.1 & \text{if } p \in \text{Background} \ . \end{cases} \tag{3.7}$$

FastFlow3D's architecture is a PointPillars-style encoder [124], traditionally used for efficient LiDAR object detection [139], that converts the point cloud into a birds-eye-view pseudoimage using infinitely tall voxels (pillars). This pseudoimage is then processed with a 4 layer U-Net style backbone. The encoder of the U-Net processes the P_t and P_{t+1} pseudoimage separately, and the decoder jointly processes both pseudoimages. A small MLP is used to decode flow for each point in P_t using the point's coordinate and its associated pseudoimage feature.

As discussed in Section 3.7.1, FastFlow3D's architectural design choices make fast even on full-size point clouds. While most feedforward methods are evaluated using a standard toy evaluation protocol with subsampled point clouds, FastFlow3D is able to scale up to full resolution point clouds while maintaining real-time performance and emitting competitive quality scene flow estimates using human supervision, making it a good candidate for the distillation step of SFvD.

In order to train FastFlow3D using pseudo-labels, we replace the foreground / background scaling function (Equation 3.7) with a simple uniform weighting ($\sigma(\cdot) = 1$), which collapses to Average EPE. Additionally, we depart from FastFlow3D's problem setup in two minor ways: we delete ground points using dataset provided maps, a standard pre-processing step [119], and use the standard scene flow problem setup of predicting flow between two frames instead of predicting future flow vectors in meters per second

In order to take advantage of the unlabeled data scaling of SFvD, we expand FastFlow3D to a family of models by designing a higher capacity backbone, producing *FastFlow3D XL*. This larger backbone halves the size of each pillar to quadruple the pseudoimage area, doubles the size of the pillar embedding, and adds an additional layer to maintain the network's receptive field in metric space; as a result, the total parameter count increases from 6.8 million to 110 million.

3.8 Experiments

ZeroFlow provides a family of fast, high quality scene flow estimators. In order to validate this family and understand the impact of components in the underlying Scene Flow via Distillation framework, we perform extensive experiments on the Argoverse 2 [144] and Waymo Open [138] datasets. We compare to author implementations of NSFP [127] and [119], implement FastFlow3D [122] ourselves (no author implementation is available), and use [119]'s implementations for all other baselines.

As discussed in [119], downstream applications typically rely on good quality scene flow estimates for foreground points. Most scene flow methods are evaluated using average Endpoint Error; however, roughly 80% of real-world point clouds are background, causing average EPE to be dominated by background point performance. To address this, we use the improved evaluation metric proposed by [119], *Threeway EPE*.

3.8.1 How does ZeroFlow perform compared to prior art on real point clouds?

The overarching promise of ZeroFlow is the ability to build fast, high quality scene flow estimators that improve with the the availability of large-scale *unlabeled* data. Does ZeroFlow deliver on this promise? How does it compare to state-of-the-art methods?

To characterize the ZeroFlow family's performance, we use Argoverse 2 to perform scaling experiments along two axes: dataset size and student size. For our standard size configuration, we use the Argoverse 2 Sensor *train* split and the standard FastFlow3D architecture, enabling head-to-head comparisons against the fully supervised FastFlow3D as well as other baseline methods. For our scaled up dataset (denoted *3X* in all experiments), we use the Argoverse 2 Sensor *train* split and concatenate a roughly twice as large set of unannotated frame pairs from the Argoverse 2 LiDAR dataset, uniformly sampled from its 20,000 sequences to maximize data diversity. For our scaled up student architecture (denoted *XL* in all experiments), we use the XL backbone described in Section 3.7.3.

Table 3.6: Quantitative results on the Argoverse 2 Sensor validation split using the evaluation protocol from [119]. The methods used in this paper, shown in the first two blocks of the table, are trained and evaluated on point clouds within a 102.4m × 102.4m area centered around the ego vehicle (the settings for the *Argoverse 2 Self-Supervised Scene Flow Challenge*) . However, following the protocol of [119], all methods report error on points in the 70m × 70m area centered around the ego vehicle. Runtimes are collected on an NVIDIA V100 with a batch size of 1 [135]. FastFlow3D, ZeroFlow 1X, and ZeroFlow 3X have identical feedforward architectures and thus share the same real-time runtime; FastFlow3D XL, ZeroFlow XL 1X, and ZeroFlow XL 3X have identical feedforward architectures and thus share the same runtime. Methods with an * have performance averaged over 3 training runs. Underlined methods require human supervision.

	Runtime (ms)		Point Cloud Subsampled Size	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE
FastFlow3D* [122]			Full Point Cloud	0.071	0.186	0.021	0.006
ZeroFlow 1X* (Ours)	29.33±	2.38	Full Point Cloud	0.088	0.231	0.022	0.011
ZeroFlow 3X (Ours)			Full Point Cloud	0.064	0.164	0.017	0.011
ZeroFlow 5X (Ours)			Full Point Cloud	**0.056**	0.140	0.017	0.011
FastFlow3D XL			Full Point Cloud	0.055	0.139	0.018	0.007
ZeroFlow XL 1X (Ours)	260.61±	1.21	Full Point Cloud	0.070	0.178	0.019	0.013
ZeroFlow XL 3X (Ours)			Full Point Cloud	**0.054**	0.131	0.018	0.012
NSFP w/ Motion Comp [127]	26,285.0±	18,139.3	Full Point Cloud	0.067	0.131	0.036	0.034
Chodosh et al. [119]	35,281.4±	20,247.7	Full Point Cloud	0.055	0.129	0.028	0.008
Odometry	—		Full Point Cloud	0.198	0.583	0.010	0.000
ICP [61]	523.11±	169.34	Full Point Cloud	0.204	0.557	0.025	0.028
Gojcic [70]	6,087.87±	1,690.56	20000	0.083	0.155	0.064	0.032
Sim2Real [79]	99.35±	13.88	8192	0.157	0.229	0.106	0.137
EgoFlow [106]	2,116.34±	292.32	8192	0.205	0.447	0.079	0.090
PPWC [113]	79.43±	2.20	8192	0.130	0.168	0.092	0.129
FlowStep3D [83]	687.54±	3.13	8192	0.161	0.173	0.132	0.176

As shown in Table 3.6, ZeroFlow is able to leverage scale to deliver superior

performance. While ZeroFlow 1X loses a head-to-head competition against the human-supervised FastFlow3D on both Argoverse 2 (Table 3.6) and Waymo Open (Table 3.7), scaling the distillation process to additional unlabeled data provided by Argoverse 2 enables ZeroFlow 3X to significantly surpass the performance of both methods just by training on more pseudo-labled data. ZeroFlow 3X even surpasses the performance of its own teacher, NSFP, *while running in real-time!*

ZeroFlow's pipeline also benefits from scaling up the student architecture. We modify ZeroFlow's architecture with the much larger XL backbone, and show that our ZeroFlow XL 3X is able to combine the power of dataset and model scale to outperform all other methods, including significantly outperform its own teacher. Our simple approach achieves **state-of-the-art** on both the Argoverse 2 validation split and ***Argoverse 2 Self-Supervised Scene Flow Challenge***.

Table 3.7: Quantitative results on Waymo Open using the evaluation protocol from [119]. Runtimes are scaled to approximate the performance on a V100 [125]. Both FastFlow3D and ZeroFlow 1X have identical feedforward architectures and thus share the same runtime. Underlined methods require human supervision.

	Runtime (ms)		Point Cloud Subsampled Size	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE
ZeroFlow 1X (Ours)	21.66±	0.48	Full Point Cloud	0.092	0.216	0.015	0.045
FastFlow3D [122]			Full Point Cloud	0.078	0.195	0.015	0.024
Chodosh [119]	93,752.3±	76,786.1	Full Point Cloud	**0.041**	0.073	0.013	0.039
NSFP [127]	90,999.1±	74,034.9	Full Point Cloud	0.100	0.171	0.022	0.108
ICP [61]	302.70±	157.61	Full Point Cloud	0.192	0.498	0.022	0.055
Gojcic [70]	501.69±	54.63	20000	0.059	0.107	0.045	0.025
EgoFlow [106]	893.68±	86.55	8192	0.183	0.390	0.069	0.089
Sim2Real [79]	72.84±	14.79	8192	0.166	0.198	0.099	0.201
PPWC [113]	101.43±	5.48	8192	0.132	0.180	0.075	0.142
FlowStep3D [83]	872.02±	6.24	8192	0.169	0.152	0.123	0.232

3.8.2 How does ZeroFlow scale?

Section 3.8.1 demonstrates that ZeroFlow can leverage scale to capture state-of-the-art performance. However, it's difficult to perform extensive model tuning for large training runs, so predictable estimates of performance as a function of dataset size are critical [134]. Does ZeroFlow's performance follow predictable scaling laws?

We train ZeroFlow and FastFlow3D on sequence subsets / supersets of the Argoverse 2 Sensor train split. Figure 3.12 shows ZeroFlow and FastFlow3D's validation Threeway EPE both decrease roughly logarithmically, and this trend appears to hold for XL backbone models as well.

Empirically, ZeroFlow adheres to predictable scaling laws that demonstrate more data (and more parameters) are all you need to get better performance. This makes ZeroFlow a practical pipeline for building *scene flow foundation models* [117] using the raw point cloud data that exists *today* in the deployment logs of Autonomous Vehicles and other deployed systems.

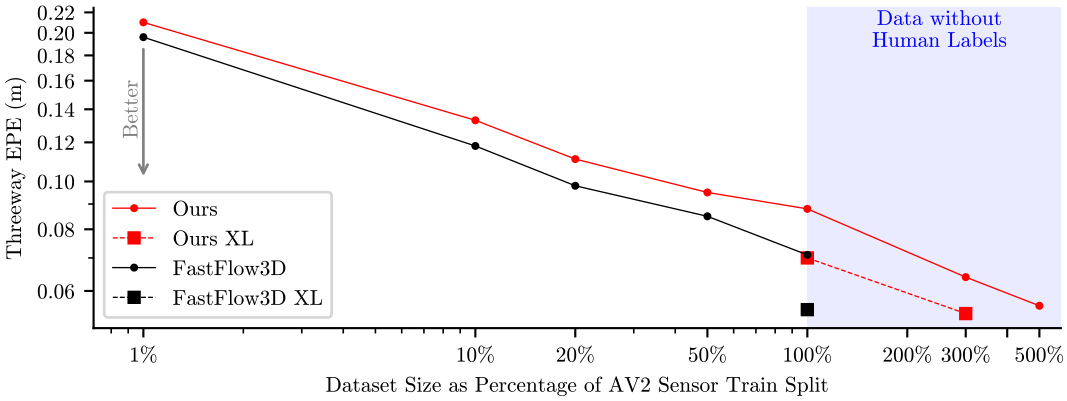


Figure 3.12: Empirical scaling laws for ZeroFlow. We report Argoverse 2 validation split Threeway EPE as a percentage of the Argoverse 2 *train* split used, on a $\log_{10}$-$\log_{10}$ scale, trained to convergence. Threeway EPE performance of ZeroFlow scales logarithmically with the amount of training data.

3.8.3 How does dataset diversity influence ZeroFlow's performance?

In typical human annotation setups, a point cloud *sequence* is given to the human annotator. The human generates box annotations in the first frame, and then updates the pose of those boxes as the objects move through the sequence, introducing and removing annotations as needed. This process is much more efficient than annotating disjoint frame pairs, as it amortizes the time spent annotating most objects in the sequence. This is why most human annotated training datasets (e.g. Argoverse 2 Sensor, Waymo Open) are composed of contiguous *sequences*. However, contiguous frames have significant structural similarity; in the 150 frames (15 seconds) of an Argoverse 2 Sensor sequence, the vehicle typically observes no more than a city block's worth of unique structure. ZeroFlow, which requires *zero* human labels, does not have this constraint on its pseudo-labels; NSFP run on non-sequential frames is no more expensive than NSFP run on non-sequential frames, enabling ZeroFlow to train on a more diverse dataset. How does dataset diversity impact performance?

To understand the impact of data diversity, we train a version of ZeroFlow 1X and ZeroFlow 2X *only* on the diverse subset of our Argoverse 2 LiDAR data selected by uniformly sampling 12 frame pairs from each of the 20,000 unique sequences (Table 3.8).

Dataset diversity has a non-trivial impact on performance; ZeroFlow, by virtue of being able to learn across *non-contiguous* frame pairs, is able to see more unique scene structure and thus learn to better to extract motion in the presence of the unique geometries of the real world.

Table 3.8: Comparison between ZeroFlow trained on Argoverse 2 Sensor dataset versus the more diverse, unlabeled Argoverse 2 LiDAR subset described in Section 3.8.1. Diverse training datasets result in non-trivial performance improvements.

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE
FastFlow3D* [122]	0.071	0.186	0.021	0.006
ZeroFlow 1X (AV2 Sensor Data)*	0.088	0.231	0.022	0.011
ZeroFlow 1X (AV2 LiDAR Subset Data)	0.082	0.218	0.018	0.009
ZeroFlow 2X (AV2 LiDAR Subset Data)	0.072	0.184	0.022	0.011

3.8.4 How do the noise characteristics of ZeroFlow compare to other methods?

ZeroFlow distills NSFP into a feedforward model from the FastFlow3D family. Section 3.8.1 highlights the *average* performance of ZeroFlow across Threeway EPE catagories, but what does the error *distribution* look like?

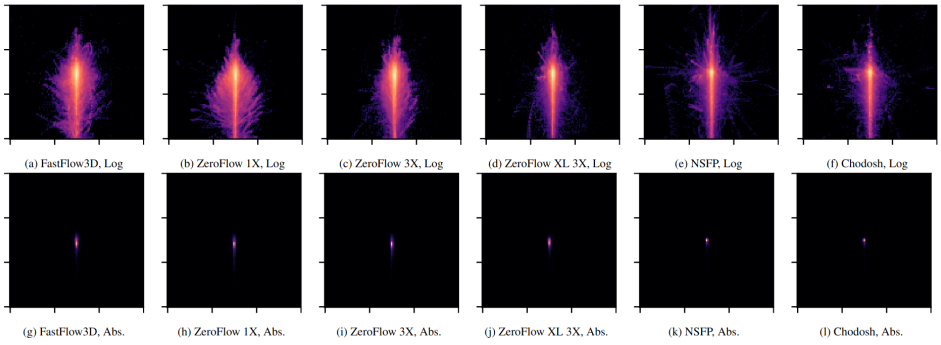


Figure 3.13: Normalized frame birds-eye-view heatmaps of endpoint residuals for Chamfer Distance, as well as the outputs for NSFP and Chodosh on moving points (points with ground truth speed above 0.5m/s). Perfect predictions would produce a single central dot. Top row shows the frequency on a $\log_{10}$ color scale, bottom row shows the frequency on an absolute color scale. Qualitatively, methods with better quantitative results have tighter residual distributions.

To answer this question, we plot birds-eye-view flow vector residuals of NSFP, Chodosh, FastFlow3D, and several members of the ZeroFlow family on moving objects from the Argoverse 2 validation dataset, where the ground truth is rotated vertically and centered at the origin to present all vectors in the same frame. Qualitatively, these plots show that error is mostly distributed along the camera ray and distributional tightness ($\log_{10}$ plots) roughly corresponds to overall method performance.

Overall, these plots provide useful insights to practitioners and researchers, par-

ticularly for consumption in downstream tasks; as an example, open world object extraction [133] requires the ability to threshold for motion and cluster motion vectors together to extract the entire object. Decreased average EPE is useful for this task, but understanding the magnitude and *distribution* of flow vectors is needed to craft good extraction heuristics.

3.8.5 How does teacher quality impact ZeroFlow's performance?

As shown in Section 3.8.1 [119] has superior Threeway EPE over NSFP on both Argoverse 2 and Waymo Open. Can a better performing teacher lead a better version of ZeroFlow?

To understand the impact of a better teacher, we train ZeroFlow on Argoverse 2 using superior quality flow vectors from [119], which proposes a refinement step to NSFP lablels to provide improvements to flow vector quality (Table 3.9). ZeroFlow trained on Chodosh refined pseudo-labels provides no meaningful quality improvement over NSFP pseudo-labels. These results also hold for our ablated speed scaled version of ZeroFlow as well.

Since increasing the quality of the teacher over NSFP provides no noticeable benefit, can we get away with using a significantly faster but lower quality teacher to replace NSFP, e.g. the commonly used self-supervised proxy of TruncatedChamfer?

To understand if NSFP is necessary, we train ZeroFlow on Argoverse 2 using pseudo-labels from the nearest neighbor, truncated to 2 meters as with TruncatedChamfer. ZeroFlow trained on TruncatedChamfer pseudo-labels performs significantly worse than NSFP, motivating the use of NSFP as a teacher.

Table 3.9: Comparison between ZeroFlow trained on Argoverse 2 using NSFP pseudo-labels, ZeroFlow using [119] pseudo-labels, and ZeroFlow using TruncatedChamfer. Methods with an * have performance averaged over 3 training runs. The minor quality improvement of Chodosh pseudo-labels does not lead to a meaningful difference in performance, while the significant degradation of TruncatedChamfer leads to significantly worse performance.

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE
ZeroFlow 1X (NSFP pseudo-labels)*	0.088	0.231	0.022	0.011
ZeroFlow 1X ([119] pseudo-labels)	0.085	0.234	0.018	0.004
ZeroFlow 1X (TruncatedChamfer pseudo-labels)	0.105	0.226	0.049	0.040

Chapter 4

Towards Multi-Signal Recovery

4.1 Introduction

Dynamic scene understanding aims to produce a model of the world that explains all measurements over time. In the context of depth sensors, this problem is posed as dynamic surface reconstruction, where the goal is to produce a time-varying surface that matches a sequence of depth measurements. This problem has been widely studied in the context of handheld RGB-D sensors capturing human-scale scenes[161, 165, 179, 175]. However, investment in autonomous driving has created a new mode of depth capture — spinning LiDAR sensors atop moving vehicles — which is largely unaddressed by the existing research. Existing methods focus on reconstructing a few densely-scanned non-rigid objects, but autonomous driving scenes are typically composed of many sparsely-scanned rigid objects [70, 149]. In this work, we propose the first dynamic surface reconstruction system aimed at operating in this setting. In addition to producing compelling visual results, our system is able to **substantially improve the quality of ground truth annotations** of ego-vehicle pose and object tracks provided in flagship datasets such as NuScenes [60] and Argoverse[174].

Approach: We address the dynamic scene reconstruction problem from a classic "analysis by synthesis" perspective; we synthesize a dense spacetime reconstruction via a compositional model of geometry and motion. We then measure the 3D error of the reconstruction with respect to the observed LiDAR scans. Finally, we optimize the geometry and motion to minimize this 3D error. We take care to formulate the optimization so that it can be efficiently decomposed into alternating steps of 1) estimating 6-DOF motion parameters of rigidly-moving components (including the moving ego-vehicle) and 2) estimating the geometry of each rigid component (including the static background). Such a decomposition allows us to leverage off-the-shelf solutions to the point registration and point-to-mesh surface reconstruction problems, respectively. Interestingly, by modeling 6-DOF pose trajectories continuously, our reconstructions can easily account for the "rolling shutter" effects of rotating

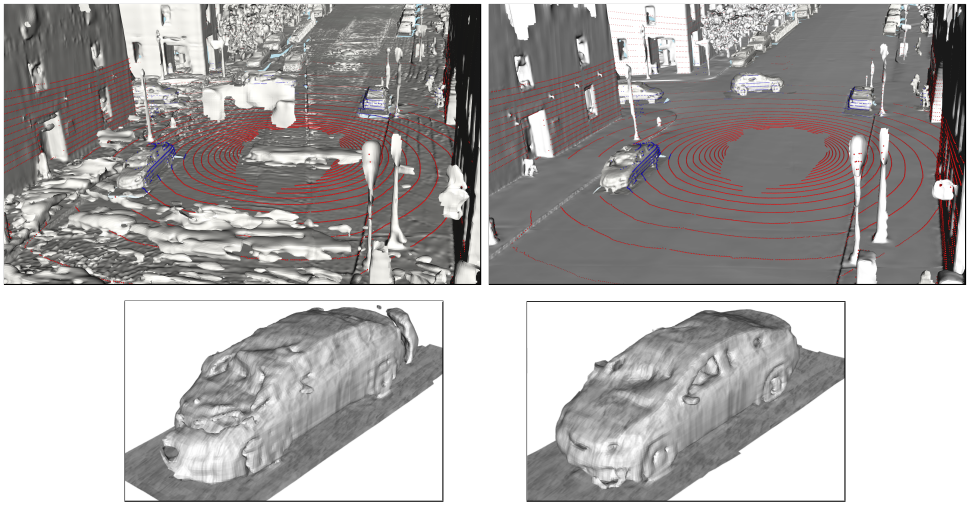


Figure 4.1: Surface reconstruction of a dynamic sequence from NuScenes. Given ego-pose and bounding box annotations, a naive approach would aggregate background and object points into common reference frames and then run a point-to-surface reconstruction algorithm. Even human annotations are not accurate enough for this simple approach (**top left**). Instead, we design an optimization that refines both the ego and object poses, yielding high-quality reconstructions (**top right**). The input LiDAR sweep is plotted with red and blue spheres, red for background points and blue for dynamic. The naive approach also fails due to rolling shutter effects on fast-moving vehicles (**bottom left**) which we correct for (**bottom right**)

LiDAR scanners. This allows our reconstructions to properly motion-compensate LiDAR scans for moving objects (for the first time, to our knowledge), complementing widely-used techniques for motion-compensation of static scenes.

Applications: Our goal is to generate dynamic scene reconstructions that provide high-quality annotations for downstream autonomous driving tasks. Labeling in-the-wild data is extremely costly, and as a result, many autonomous driving tasks rely on re-processing existing data of varying quality. For example, depth completion benchmarks use aggregated LiDAR sweeps to generate ground truth "dense" depth reconstructions [169]. This results in annotated data with well-documented occlusion errors and motion artifacts that are nonetheless still used for training and evaluation [173, 180]. An example of the depth maps produced by our method is shown in Fig. 4.3. Scene flow is another autonomous driving task that re-processes existing AV datasets, using annotated bounding box motion between frames as a proxy for the underlying ground-truth motion field [57, 87], which also has well-documented issues in evaluation [149]. Accurate time-space reconstructions of the rigidly moving objects in the scene are critical to both of these tasks. We demonstrate that the ground-truth motion annotations are insufficient for producing these reconstructions and that our system significantly improves upon them. We provide

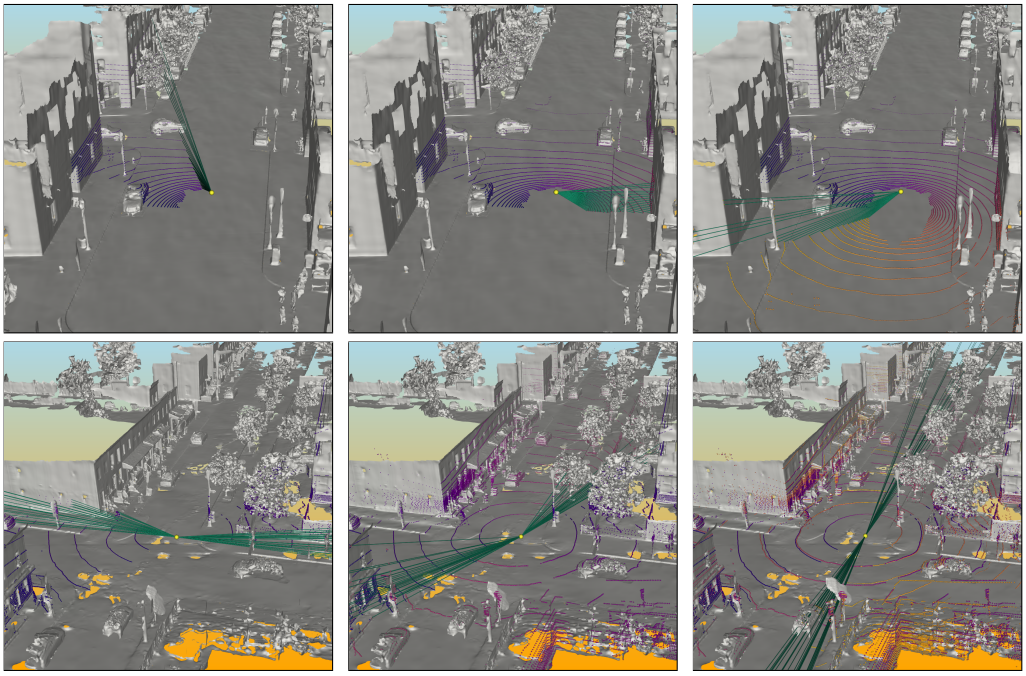


Figure 4.2: LiDAR returns are often grouped and processed in 360-degree sweeps, but most sensors have a continuous "shutter" as they rotate. Our framework can model this continuous shutter for any combination of LiDAR sensors or scanning patterns. We find that this modeling has a large impact on the quality of reconstructed objects. Here, we visualize the set of points within a sweep, which are captured simultaneously (green lines) for NuScenes (**top**) and Argoverse (**bottom**). Argoverse has two LiDAR sensors spinning 180 degrees out of phase, leading to two sets of points being captured at each instant.

numerous qualitative visuals that speak to our accuracy (including those in the supplement), but providing quantitative results is challenging since we often outperform the ground truth to which one normally compares! That said, we do provide quantitative metrics such as point-to-surface error metrics. Moreover, we show that our dynamic reconstruction engine can already be used as a practical system for fully or semi-automatic annotation by converting the output of off-the-shelf object trackers or low-frame rate human annotations into high-frame rate reconstructions. In particular, we outperform the baseline approach of linear-interpolated annotations, which is widely-used despite its simplicity [57, 87, 149].

In short, our main contributions are posing the classic dynamic surface reconstruction problem in a new setting, proposing new downstream applications of this problem, and demonstrating a simple yet effective optimization-based solution.

4.2 Related Work

Dynamic Surface Reconstruction: Reconstruction of non-rigid surfaces from depth scanners has been studied for over two decades[160]. Early work overcame the inherent ill-posedness of the problem by relying on object-specific shape models for humans[166], faces[157] and hands[164]. Since then, many works have demonstrated template-free reconstruction in both the online[161] and offline settings[162]. This line of work is focused on highly deformable objects such as people and animals, which are very close to the depth sensor. As a result, they do not apply to the long-range, generally rigid world of autonomous driving scenes.

Dynamic SLAM: Since we are solving for the global map of the world as well as the sensor's location within it, our work is closely related to SLAM in general and specifically to Dynamic SLAM, sometimes called SLOT (Simultaneous Localization and Object Tracking). Many works identify dynamic objects to remove them from the global map[152], but some track dynamic objects and register new observations to an object template. Similar to our work, these approaches typically represent the world as a composition of rigid bodies[148, 171, 151, 176, 153, 167]. These methods are focused on real-time operation from RGB inputs rather than offline LiDAR processing. As a result, they generally do not reconstruct detailed surface representations of the tracked objects, although that has been proposed as a post-processing step to the tracked objects[155]. Also similar to our work are SLAM methods, which create a dense surface reconstruction of the global map[170, 157]. However, to our knowledge, none of these approaches reconstruct dynamic objects.

Asset Generation for Autonomous Driving: Related to the object reconstruction component of our system is the line of work focused on creating high-quality mesh reconstructions of vehicles for simulation purposes[177, 158, 172]. These methods are similar to ours in that they reconstruct dense meshes of in-the-wild vehicles but have several key differences. First, since these systems aim to extract assets, not reconstruct complete sequences, they focus on objects that are close to the sensor and have accurate poses from object detection. Although this is not made explicit, the result is that these systems are made to operate on stationary objects, not dynamic ones. Second, they rely heavily on RGB information as well as depth sensors. As we show, reconstructing moving objects from spinning LiDARs requires careful handling of the rolling-shutter effect. This makes it challenging to incorporate global-shutter RGB cameras. The fact that these works make no mention of this further indicates that they do not handle dynamic objects.

4.3 Problem Statement

We assume as input a sequence of LiDAR sweeps measured at timestamps $t \in \mathcal{T}$, and coarse tracks of K objects. Since we are using a compositional model of the scene, we will need a coordinate frame for each component.

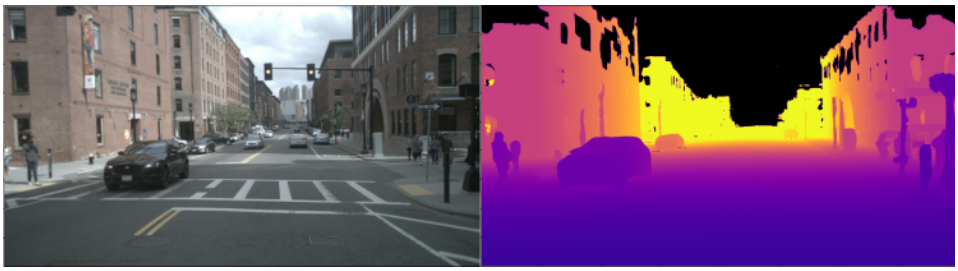


Figure 4.3: An example dense depth map produced by our method

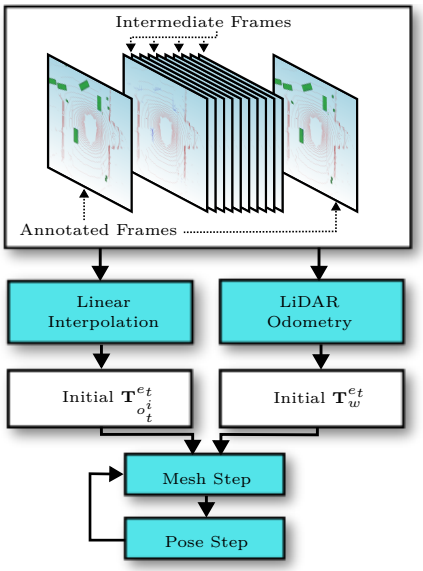


Figure 4.4: A high-level overview of our method when used with sparse ground truth annotations. We take the annotated LiDAR frames that make use of interpolation and off-the-shelf LiDAR odometry to initialize object and ego poses for all frames. Our global optimization makes use of coordinate descent to update the geometry and motion alternatingly. When using the output of an object tracker as input, we omit the interpolation step, as the tracks cover all the input frames.

- **Ego coordinates:** This is the coordinate frame that the input points are measured in. That is, the coordinate frame where the LiDAR sensor is at the origin and the z-direction points along the rotation axis. Since the ego-vehicle is moving, this coordinate frame changes over time. We will denote the sensor coordinate frame at time t as e_t.

- **Object coordinates:** To each of the K objects in the sequence, we will assign a coordinate system where the object is at the origin, the z-direction is up and the x-direction is "forward". Each of these coordinate systems also varies with

time to express the dynamic object motion. We will denote the ith object's coordinate frame at time t as o_t^i.

- **World coordinates:** This is the fixed global coordinate frame of the scene, which we denote as w. Importantly, we represent the static background in this fixed world coordinate frame. Due to the global coordinate frame ambiguity, we will choose this frame to be equal to e_1.

To indicate the coordinate frame of given point $\mathbf{x}$, or set of points $\mathbf{X}$ we will use subscripts: for example, we write input points as $\mathbf{x}_{e_t}$, $\mathbf{X}_{e_t}$. We will express the relationships between these coordinate frames using 4×4 rigid transformation matrices $\mathbf{T}$. We write the transformation from world coordinates at time t to sensor coordinates e_t as $\mathbf{T}_w^{e_t}$. Similarly, the transformation from object i at time t to world coordinates is written as $\mathbf{T}_{o_t^i}^w$. Then, transformation from the ith object's coordinate system at time t to the sensor coordinates at time t can be written as $\mathbf{T}_{o_t^i}^{e_t} = \mathbf{T}_w^{e_t}\mathbf{T}_{o_t^i}^w$.

We aim to decompose the scene into a set of surfaces that transform rigidly over time. Our approach is agnostic to the particular choice of surface representation, but we use triangular meshes since they are lightweight and widely used. We will have a mesh for each of the K objects in the scene $\{\mathcal{M}_i\}_{i=1}^K$ as well as the background $\mathcal{M}_0$. In a slight abuse of notation, we will write $\mathbf{T}\mathcal{M}$ to denote transforming the vertices of $\mathcal{M}$ by the transformation $\mathbf{T}$. Similarly, we will write $\mathbf{TX}$ to express transforming the points $\mathbf{X}$. The union of two meshes will be written as $[\mathcal{M}_1, \mathcal{M}_2]$. Finally, we will measure the 3D distance between a mesh and a point cloud using the nearest neighbor loss

$$\mathcal{D}(\mathcal{M}, \mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} \min_{\mathbf{m} \in \mathcal{M}} \|\mathbf{m} - \mathbf{x}\|. \tag{4.1}$$

4.4 Objective

We aim to find surfaces and their 6-DOF motion parameters such that their composition matches the measured pointcloud at each timestep. Since our pointclouds are measured in the ego coordinates e_t, we must transform our meshes into that frame. Consider a scene composed of a background mesh ($\mathcal{M}_0$) and a single object ($\mathcal{M}_1$). To transform object $\mathcal{M}_1$ into e_t, we first place it in the world via $\mathbf{T}_{o_t^1}^w$ and then view the world from the sensor frame via $\mathbf{T}_w^{e_t}$. We can use the transformation $\mathbf{T}_{o_1^1}^{e_t} = \mathbf{T}_w^{e_t}\mathbf{T}_{o_t^1}^w$ to accomplish both. To transform the static background mesh (which is already represented in world coordinates), we need only transform it by $\mathbf{T}_w^{e_t}$. Once all surfaces have been transformed into frame e_t, the composite reconstruction $\left[\mathbf{T}_w^{e_t}\mathcal{M}_0, \mathbf{T}_{o_1^t}^{e_t}\mathcal{M}_1\right]$ is compared to the measured LiDAR points $\mathbf{X}_{e_1}$ using the nearest-neighbor distance from eq. (4.1). Summing this over all time produces

our final reconstruction error:

$$\min_{\{\mathcal{M}_i, \mathbf{T}_{o_t^i}^{e_t}, \mathbf{T}_w^{e_t}\}} \sum_{t \in \mathcal{T}} \mathcal{D}(\left[\mathbf{T}_w^{e_t}\mathcal{M}_0, \mathbf{T}_{o_t^1}^{e_t}\mathcal{M}_1, \ldots, \mathbf{T}_{o_t^K}^{e_t}\mathcal{M}_K\right], \mathbf{X}_{e_t}). \tag{4.2}$$

4.4.1 Decomposition

We could use a differentiable renderer and optimize eq. (4.2) with gradient descent. But, we will demonstrate that decomposing the optimization into discrete sub-components allows us to leverage off-the-shelf tools and yields good reconstructions. To aid in this decomposition, let $\mathbf{X}_{e_t}^i$ denote the subset of points from $\mathbf{X}_{e_t}$ which fall on object i. This assignment can be coarse, and in practice, we assign points to the bounding box they fall into and to the background if they are not contained in any bounding box. Once we have refined the poses of the bounding boxes, we can recompute this step to get new assignments. Using this notation, we can further break down eq. (4.2) into:

$$\min_{\{\mathcal{M}_i, \mathbf{T}_{o_t^i}^{e_t}, \mathbf{T}_w^{e_t}\}} \sum_{t \in \mathcal{T}} \sum_{i=0}^{K} \mathcal{D}(\mathbf{T}_{o_t^i}^{e_t}\mathcal{M}_i, \mathbf{X}_{e_t}^i), \tag{4.3}$$

where we let $o_t^0 = w$ for notional simplicity.

Our approach consists of applying coordinate descent: alternating between fixing the poses to optimize the meshes and then fixing the meshes to update the poses. These stages are the **pose step** and **mesh step**, respectively. The coarse bounding boxes are used to initialize $\mathbf{T}_{o_t^i}^{e_t}$ and an off-the-shelf LiDAR odometry method is used to initialize $\mathbf{T}_w^{e_t}$. We do not require any initialization of the meshes. A schematic of the pipeline is shown in Fig. 4.4.

4.4.2 Mesh Step

Assuming fixed poses, we can estimate new meshes by solving

$$\mathcal{M}_i \leftarrow\!\shortmid \arg\min_{\mathcal{M}_i} \sum_{t \in \mathcal{T}} \mathcal{D}(\mathbf{T}_{o_t^i}^{e_t}\mathcal{M}_i, \mathbf{X}_{e_t}^i). \tag{4.4}$$

We can make use of two identities related to the nearest neighbor distance to transform this optimization into a well-known problem. First, we can use the fact the distance is unaffected by a global rigid transformation to see that $\mathcal{D}(\mathbf{T}\mathcal{M}, \mathbf{X}) = \mathcal{D}(\mathcal{M}, \mathbf{T}^{-1}\mathbf{X})$. Second, if we write a set of points $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2]$ as a union of two disjoint sets $\mathbf{X}_1$ and $\mathbf{X}_2$, we can see that $\mathcal{D}(\mathcal{M}, [\mathbf{X}_1, \mathbf{X}_2]) = \mathcal{D}(\mathcal{M}, \mathbf{X}_1) + \mathcal{D}(\mathcal{M}, \mathbf{X}_2)$.

Now we combine them to get:

$$\begin{aligned}\mathcal{M}_i &\leftarrow\!\shortmid \arg\min_{\mathcal{M}_i} \sum_{t\in\mathcal{T}} \mathcal{D}(\mathbf{T}^{e_t}_{o^i_t}\mathcal{M}_i, \mathbf{X}^i_{e_t}) \\ &= \arg\min_{\mathcal{M}_i} \sum_{t\in\mathcal{T}} \mathcal{D}(\mathcal{M}_i, (\mathbf{T}^{e_t}_{o^i_t})^{-1}\mathbf{X}^i_{e_t}) \\ &= \arg\min_{\mathcal{M}_i} \mathcal{D}\left(\mathcal{M}_i, \left[(\mathbf{T}^{e_t}_{o^i_t})^{-1}\mathbf{X}^i_{e_t}, \ldots\right]\right).\end{aligned} \tag{4.5}$$

The final form of this equation can be interpreted as a standard static point-to-surface reconstruction problem. We use the recent Neural Kernel Surface Reconstruction [154], but any technique, such as Poisson surface reconstruction [156], could be used.

4.4.3 Pose Step

Assuming fixed meshes, we can estimate new poses by solving

$$\begin{aligned}\mathbf{T}^{e_t}_{o^i_t} &\leftarrow\!\shortmid \arg\min_{\mathbf{T}^{e_t}_{o^i_t}} \mathcal{D}\left(\mathbf{T}^{e_t}_{o^i_t}\mathcal{M}_i, \mathbf{X}^i_{e_t}\right) \\ &= \arg\min_{\mathbf{T}^{e_t}_{o^i_t}} \mathcal{D}\left(\mathcal{M}_i, (\mathbf{T}^{e_t}_{o^i_t})^{-1}\mathbf{X}^i_{e_t}\right).\end{aligned} \tag{4.6}$$

This is a point-to-mesh registration problem that is well-studied under the family of Iterative Closest Point (ICP) methods. However, there is a complication that we thus far have avoided by being vague about the definition of a LiDAR sweep.

4.4.4 What *is* a LiDAR sweep?

Revolving LiDAR sensors do not have a global shutter. Instead, they rotate continuously and measure depth across 16-128 vertically arranged lasers, typically taking 100ms to complete a 360-degree rotation. Depth along each laser ray is measured with respect to the (potentially moving) ego sensor frame. Most software packages abstract away this continuous capture and instead generate a *virtual* sweep of point measurements that would have been obtained if the sensor captured the world a single time instant with a global 360-degree shutter. To do so, robotic platforms typically transform all points to a chosen reference frame (by exploiting knowledge of continuous ego pose during the 100ms capture window, often obtained with a constant velocity assumption).

Such motion compensation will generate the correct virtual point cloud for a static world but will **not** correctly compensate for moving objects in a dynamic world. This well-known phenomenon is often manifested as vertical "seams" that appear in a sweep since points on either side of the seam are collected 100ms apart

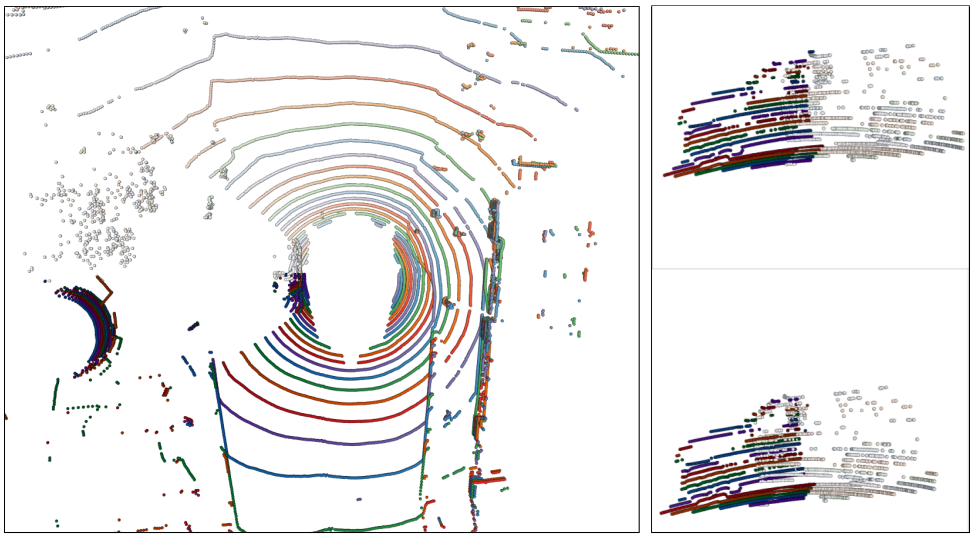


Figure 4.5: (**Left**) A LiDAR sweep where each point has been colored according to which laser it belongs to (hue) and the time within the sweep it was acquired (lighter is earlier, darker is later). A moving car is passing the ego-vehicle on the left and is captured at both the start and end of the sweep (**top right**), leading to distortion (the driver-side window is captured twice in different locations). Accounting for this distortion by modeling the object motion is key to the quality of our reconstructions (**bottom right**).

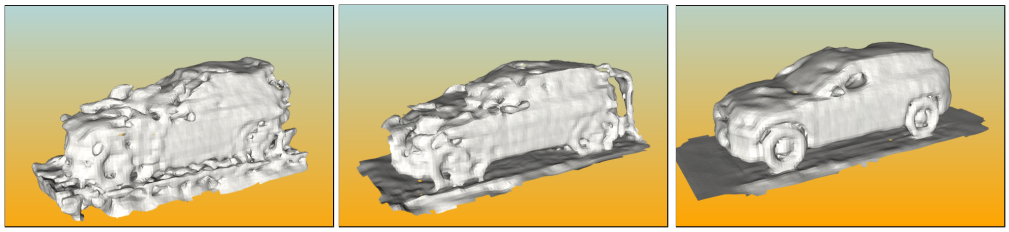


Figure 4.6: Accurate object poses and accounting for intra-sweep motion are critical for high-quality reconstructions. (**Left**) shows the reconstruction with neither refined poses nor object-motion compensation, (**middle**) shows the reconstruction with refined poses but without object-motion compensation, and (**right**) shows the result of combining both.

(Fig. 4.5). However, our spacetime optimization can correctly model moving objects by letting our time index t be a continuous variable rather than an integer frame index. For example, if we have a 16-beam LiDAR sensor that takes 1080 measurements in a single rotation, the first 16 points of our sequence are written as $\mathbf{X}_{e_{1/1080}}$. Importantly, our global optimization eq. (4.2), mesh step eq. (4.4), and pose step eq. (4.6) are just as valid under this interpretation of a sweep "slice", but with 1080 times as many poses. This is computationally expensive and may underconstrain the optimization. To avoid this, we adopt a constant velocity model for poses between "keyframes" placed at the end of every complete sensor rotation. For example, we can express the continuous pose of the sensor for $0 < t < 1$ using the keyframe poses $\mathbf{T}_{e_0}^{w}, \mathbf{T}_{e_1}^{w}$ like so:

$$\begin{aligned} \mathbf{T}_{w}^{e_1} \left(\mathbf{T}_{w}^{e_0}\right)^{-1} &= \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{v}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \quad \mathbf{w} = \log\left(\mathbf{R}\right) \\ \mathbf{T}_{e_t}^{w} = \mathbf{T}_{e_1}^{w} &\begin{bmatrix} e^{\mathbf{w}(1-t)} & \mathbf{v}(1-t) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \mathbf{T}_{e_1}^{w} \mathbf{T}_{e_t}^{e^1}. \end{aligned} \tag{4.7}$$

Just as we assumed that the ego-vehicle obeys a constant velocity model between keyframes, we can make the same assumption about the motion of other objects in the scene. However, care needs to be taken to ensure that the constant velocity assumption is applied to the object's motion *with respect to the world* as opposed to with respect to the ego-vehicle. Constant velocity in the world frame is not equivalent to constant velocity in the moving sensor frame due to the presence of rotations. With this in mind, we represent the object poses like so:

$$\begin{aligned} \left(\mathbf{T}_{e_1}^{o_1}\mathbf{T}_{w}^{e_1}\right)\left(\mathbf{T}_{e_0}^{o_0}\mathbf{T}_{w}^{e_0}\right)^{-1} &= \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{v}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \quad \mathbf{w} = \log\left(\mathbf{R}\right) \\ \mathbf{T}_{e_t}^{o_t^i} = &\begin{bmatrix} e^{\mathbf{w}(1-t)} & \mathbf{v}(1-t) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \mathbf{T}_{e_1}^{o_1^i}\mathbf{T}_{e_t}^{e_1} = \mathbf{T}_{o_1^i}^{o_t^i}\mathbf{T}_{e_1}^{o_1^i}\mathbf{T}_{e_t}^{e_1}. \end{aligned} \tag{4.8}$$

This factorization is not only the correct way of applying the constant velocity assumption but also makes it easy to deal with the fact that, in many cases, public datasets do not release the raw $\mathbf{X}_{e_t}$ points but instead release the ego-motion compensated points $\mathbf{T}_{e_t}^{e_1}\mathbf{X}_{e_t}$. With the above factorization, we can omit the first $\mathbf{T}_{e_t}^{e_1}$ transformation as it has already been applied. This is one of those fortunate situations where the easy and correct approaches are the same!

We can directly plug eq. (4.7) and eq. (4.8) into our mesh and pose steps. This is precisely what we do for the mesh step in eq. (4.4), but there is one detail to account for. Point-to-surface reconstruction methods need to disambiguate between the inside and outside of the reconstructed objects. This is often done using the empty-space constraint provided by the rays connecting the sensor to each measured point. Care needs to be taken to use the continuous sensor position defined by the continuous pose, or else the reconstruction can fail.

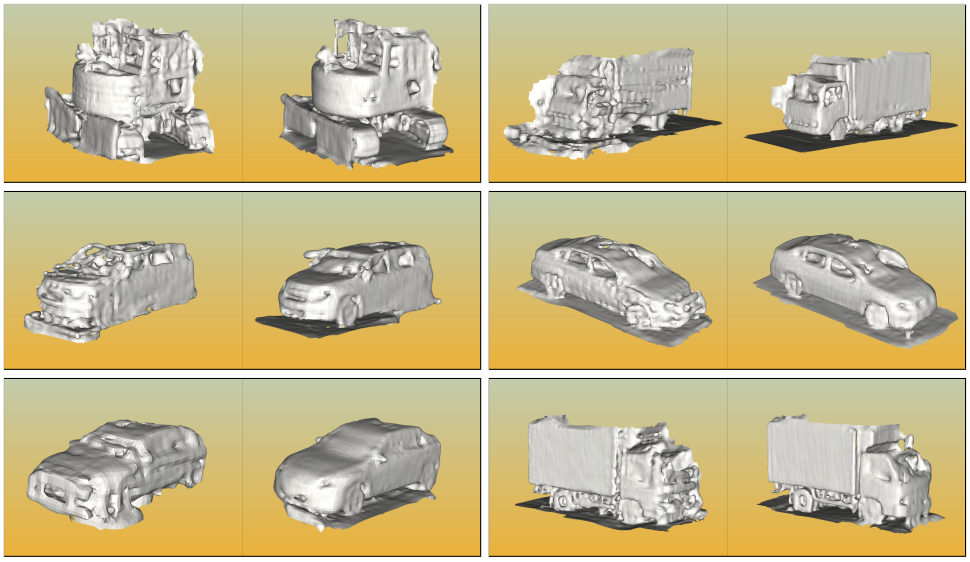


Figure 4.7: (**Left**) Each column shows NuScenes object reconstructions using ground truth poses compared to (**right**) ours.

For the pose step in eq. (4.6), naively plugging it in complicates the use of off-the-shelf ICP methods. Instead, we make an approximation where we consider the intra-sweep transformations $\mathbf{T}_{e_t}^{e_1}$ and $\mathbf{T}_{o_1^i}^{o_t^i}$ to be fixed corrections applied before estimating new keyframe poses. This corresponds to the common practice of motion-compensating a sweep, but our approach produces a 360-degree sweep that is correctly compensated *for object motion* (for the first time, to our knowledge).

4.5 Experimental Setup

We test our method on sparse LiDAR sequences from NuScenes[60] and the Argoverse 2.0[174]. NuScenes also has sparse annotations, providing them at 2Hz compared to measuring LiDAR sweeps at 20Hz. This sparsity allows us to showcase our method's ability to densify in space and time. As a result, we focus our quantitative analysis on NuScenes, but similar results for Argoverse can be found in the supplemental material. NuScenes breaks the data into 20-second sequences, each containing around 400 LiDAR sweeps. Since our method does not require any training data, we focus on qualitative and quantitative evaluation of the ten validation sequences the dataset authors chose to serve as a representative sample of the data. We omit one sequence (scene-0553) since it does not contain any motion of the vehicle. For each sequence, we initialize the ego-poses using a recent LiDAR-only odometry method[109]. We initialize the object tracks and bounding boxes either by using linear interpolation on the provided object annotations or with the output of an off-the-shelf LiDAR object tracker[163]. We then run 100 iterations of

refinement on all of the objects and background maps, with early stopping criteria to avoid wasted computation. Iterations are stopped if the mean registration error for an object falls below 1 centimeter for three consecutive iterations. For the mesh step, we use the default parameters of the publicly released Neural Kernel Surface Reconstruction model[154]. For the pose update, once we have deskewed the dynamic objects, we use a standard ICP implementation[181] with a point-to-plane loss, a robust Huber kernel with $k = 0.2$ and a matching threshold of 1.5 meters.

For the tracking results, we use the Centerpoint-based[178] object detector LT3D[163] to extract bounding boxes in all frames. We then use greedy association to turn the detections into object tracks.

4.6 Qualitative Results

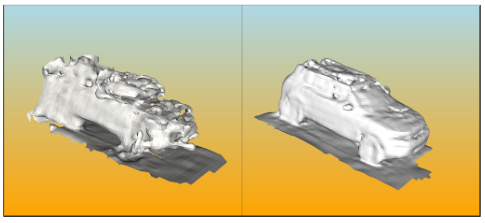


Figure 4.8: (**Left**) Argoverse object reconstructions using ground truth poses compared to (**right**) ours.

Visualizations of our foreground reconstructions on NuScenes are shown in Fig. 4.7. An example reconstruction from Argoverse is shown in Fig. 4.8. In the visualizations, we show that the ground-truth object annotations are not accurate enough to yield good reconstruction. Errors in bounding box alignment and orientation lead to point aggregation errors, leading to poor surface reconstructions. The fact that our refined object poses lead to better reconstructions is evidence that we produce better annotations than the ground truth. Motion distortion from dynamic objects also contributes to the poor quality of the ground truth reconstructions. In figure 4.6, we show how accounting for this distortion can significantly improve the reconstructions.

Background reconstructions from NuScenes are shown in Fig. 4.10 and Argoverse in Fig. 4.9. For these "objects" the quality improvement comes from refining the ego-pose of the vehicle. As with foreground objects, ego-pose errors cause misalignment of the LiDAR sweeps, which become surface artifacts. However, the comparison is with a state-of-the-art LiDAR odometry method instead of the ground truth. We use odometry because it performs better than the ground-truth ego-poses.

4.7 Quantitative Results

We analyze how well our piecewise rigid model can represent the dynamic scenes. Recent works that use volumetric scene representations[150, 168] typically use ray-casting along the ground-truth ray directions to measure this property. However, we find that this metric is dominated by large outlier errors caused by rays missing

	NN Dist (m) ↓	Acc Relax ↑	Acc Strict ↑
DSNerf[150]	0.537	0.81	0.70
SUDS[168]	0.18	0.94	0.88
NKSR[154] + GT tracks (2Hz)	0.071	0.9	0.76
NKSR[154] + LT3D[163] tracks (2Hz)	0.071	0.9	0.76
Ours + GT tracks (2 Hz)	0.048	0.96	0.91
Ours + GT tracks (1 Hz)	0.050	0.96	0.90
Ours + GT tracks (0.5 Hz)	0.048	0.96	0.91
Ours + GT tracks (0.25 Hz)	0.048	0.96	0.91
Ours + LT3D[163] tracks	0.048	0.96	0.90

Table 4.1: Surface quality evaluation on NuScenes, measured by comparing the LiDAR points to their closest points on the reconstructed surfaces.

an object boundary. This makes the metric hard to interpret since it is a mixture of two different error distributions. Instead, we propose to measure reconstruction accuracy using the nearest-neighbor distance between the input point clouds and the reconstructed scene at each timestamp. We report the average distance and two accuracy metrics to characterize the distribution of errors. Specifically, we compute the percent of points less than 10cm and 5cm for the relaxed and strict metrics, respectively.

We compare our method using different inputs (ground truth annotations with varying rates of subsampling and the output of an object tracker) to several strong baselines. First, we compare with DS-Nerf[150] and SUDS[168], two NeRF-style[159] models which have been adapted to urban scenes with LiDAR inputs. Second, we compare with the surface reconstruction method NKSR[154] combined with either ground truth object tracking or the results of the off-the-shelf tracker we use.

As seen in Tab. 4.1, our method outperforms all baselines. We find that the NeRF style methods are good at reconstructing most points, leading to high accuracy metrics, but are prone to significant outlier errors, leading to poor average error. On the other hand, the surface-based method achieves better average error but fails at reconstructing fine detail, leading to low strict accuracy. In contrast, our method produces suitable high-level geometry, leading to low average error, and faithfully reconstructs fine details, leading to high accuracy.

We use the same annotation subsampling technique to evaluate the improvement of our estimates of the locations of the dynamic objects in the scenes. By omitting input annotations, we can compare our method's predicted object locations to the ground truth using the NuScenes' Average Translation Error metric[60]. We compute this metric over the nine test sequences and filter out objects that follow linear trajectories. As shown in Tab. 4.2, our method improves the estimates of complex object motions.

	ATE (m) ↓		
	1Hz	0.5Hz	0.25Hz
Interpolation	0.29	0.40	0.74
Ours	**0.20**	**0.22**	**0.52**

Table 4.2: Pose accuracy evaluation on NuScenes (using NuScene's default ATE metric), measured by comparing the bounding box locations predicted by our method to held-out ground truth labels provided at 2Hz. We compare our method to linearly interpolating the poses as is commonly done to create scene-flow labels [57].

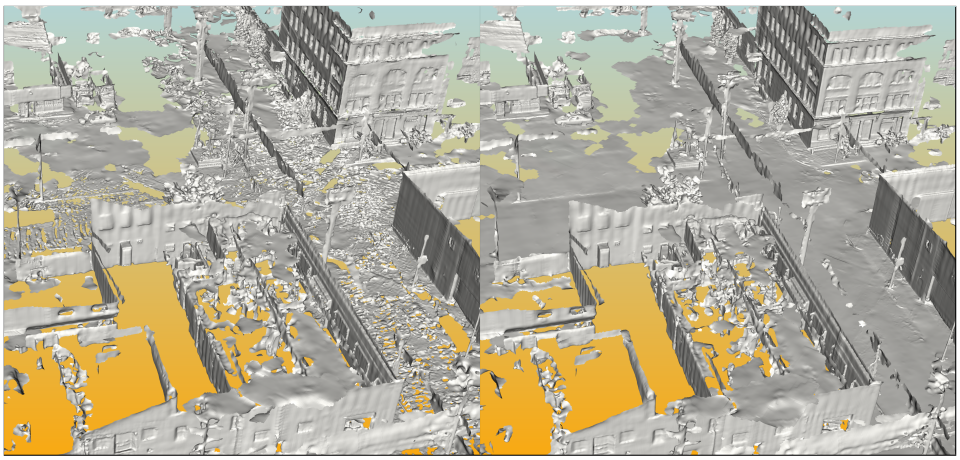


Figure 4.9: (**Left**) Map reconstruction on Argoverse 2.0 using *ground truth* poses compared to (**right**) ours.

4.8 Conclusion

In this work, we brought dense, dynamic reconstruction to the large-scale in-the-wild autonomous vehicle setting. We developed an optimization framework for understanding this problem and provided a simple yet effective solution based on decomposing the problem into well-studied sub-components. This solution yields high-quality reconstructions of both the foreground and background and can even account for subtle distortions in the input point clouds. We hope that this method will not only be useful for creating training and evaluation data for other perception tasks but will also promote active research in this challenging setting.

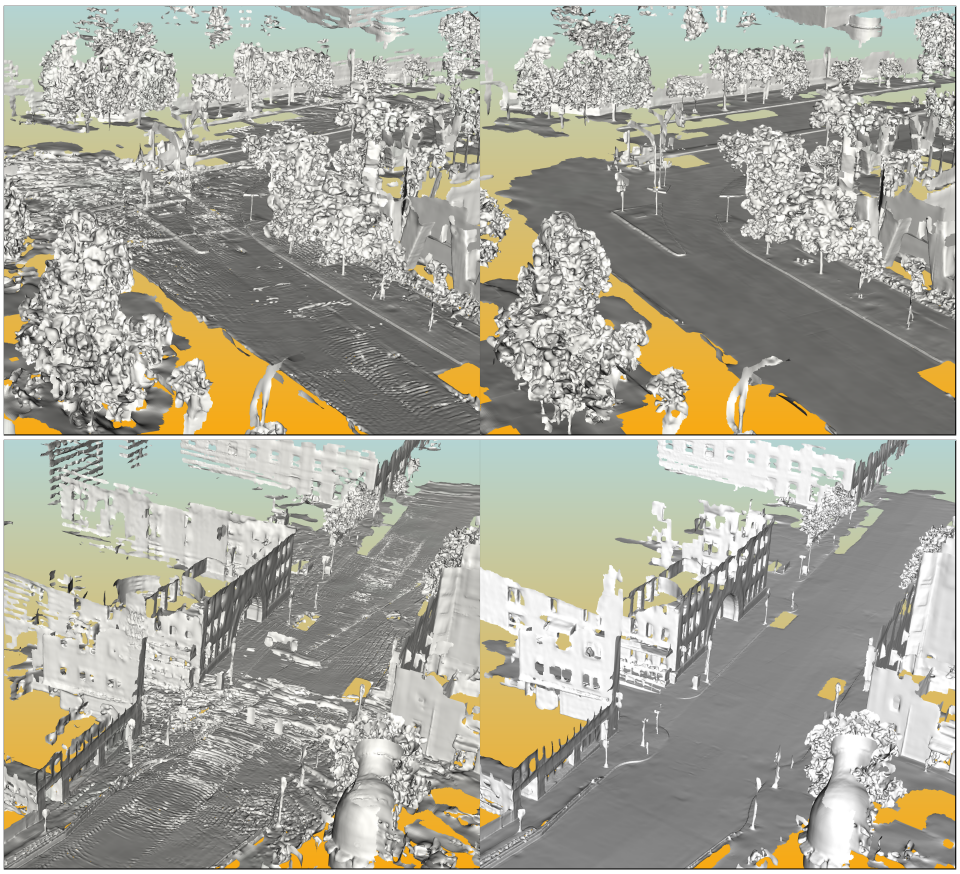


Figure 4.10: (**Left)** Map reconstructions using odometry poses compared to (**right**) ours. Ground-truth ego poses produce even worse results since NuScenes does not align poses in the z (height) dimension.

Chapter 5

Conclusions

In this thesis, we revisited the classic analysis-by-synthesis perspective on recovery problems. We aimed to bring this perspective up to date by combining test time optimizations with large-scale machine learning. In doing so, we identified several benefits of analysis-by-synthesis, the greatest of which was the ability to incorporate knowledge of the measurement process into the inference step. These ideas were explored across several tasks and settings.

The first set of tasks we considered—compression artifact removal, depth completion, and trajectory reconstruction——were all supervised by large datasets of pairs of measurements and clean signals. In this supervised setting, we designed a bi-level optimization that allowed us to learn signal models that lead to good reconstructions. We used this framework to show that more complicated forward models yield more significant gains from the test-time optimization approach.

We then examined the unsupervised recovery problem of scene flow estimation from LiDAR point clouds. We found that, without supervision, feedforward models were generally less effective than carefully designed test time optimizations. We used this insight to create a student/teacher method that distilled the information from a test time optimization into a fast feedforward network that could be scaled to huge unlabeled datasets. This distillation ultimately led to the student outperforming the teacher and achieving state-of-the-art results.

Finally, we leaned further into modeling the measurement process to estimate motion and geometry simultaneously from point clouds. In this challenging problem, we designed an analysis by synthesis optimization, which combined both signals and explored a different method for incorporating learning. Specifically, we decomposed the objective into discrete steps which could be handled by off-the-shelf networks.

Bibliography

[1] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.

[2] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[3] I. Laina, C. Rupprecht *et al.*, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 239–248.

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 21

[5] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," *arXiv preprint arXiv:1702.02706*, 2017.

[6] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," *arXiv preprint arXiv:1609.03677*, 2016.

[7] S. Hawe, M. Kleinsteuber, and K. Diepold, "Dense disparity maps from sparse disparity measurements," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2126–2133. 15

[8] L.-K. Liu, S. H. Chan, and T. Q. Nguyen, "Depth reconstruction from sparse samples: Representation, algorithm, and sampling," *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1983–1996, 2015. 15

[9] F. Ma, L. Carlone *et al.*, "Sparse sensing for resource-constrained depth reconstruction," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 96–103.

[10] F.Ma, S. Karaman, "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image", *arXiv preprint arXiv:1709.07492*, 2017 14, 16, 22

[11] M. Elad. *Sparse and Redundant Representations*. New York, NY: Springer, 2010

[12] Vardan, Papyan & Romano, Yaniv & Elad, Michael. "Convolutional Neural Networks Analyzed via Convolutional Sparse Coding". *Journal of Machine Learning Research*. 18. 2016 14

[13] C. Murdock. "Deep Component Analysis via Alternating Direction Neural Networks", *arXiv preprint*, 2018 15, 17, 19

[14] J. Uhrig, N. Schneider, L. Schneider, U. Franke, A. Geiger. "Sparsity Invariant CNNs", *arXiv preprint arXiv:1708.06500*, 2017 2, 7, 14, 16, 21, 22, 23, 24

[15] J. Sulam, V. Papyan, Y. Romano, M. Elad. "Multi-Layer Convolutional Sparse Modeling: Pursuit and Dictionary Learning". *arXiv preprint arXiv:1708.08705* 16

[16] G. Riegler, M. Ruther, H. Bischof. "ATGV-Net: Accurate Depth Super-Resolution.". in *European Conference on Computer Vision*, 2016

[17] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] N. Schneider, L. Schneider, P. Pinggera, U. Franke, M. Pollefeys, and C. Stiller. Semantically guided depth upsampling. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, pages 37–48. Springer, 2016. 22

[19] J. T. Barron and B. Poole. The fast bilateral solver. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 617–632. Springer, 2016. 22

[20] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof. Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2013. 22

[21] E. Nadaraya. On estimating regression. In *Theory of Probability and its Applications*, 1964. 22

[22] G. Watson. Smooth regression analysis. In *Sankhyā: The Indian Journal of Statistics*, 1964. 22

[23] C. Moustapha, B. Piotr, G. Edouard, D. Yann, and U. Nicolas. "Parseval networks: Imporoving robustness to adversarial examples". *arXiv preprint arXiv:1704.08847*

[24] Aviad. Aberdam, Jeremias. Sulam, and Michael. Elad. Multi-layer sparse coding: The holistic way. *SIAM Journal on Mathematics of Data Science*, 1(1):46–77, 2019. 22

[25] Michal Aharon, Michael Elad, Alfred Bruckstein, et al. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311, 2006. 19

[26] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. *CoRR*, abs/1605.00075, 2016.

[27] Nathaniel Chodosh, Chaoyang Wang, and Simon Lucey. Deep convolutional compressed sensing for lidar depth completion. *CoRR*, abs/1803.08949, 2018.

[28] Ronald R. Coifman, Yves Meyer, Steven Quake, and M. Victor Wickerhauser. Signal processing and compression with wavelet packets. *Wavelets and Their Applications*, page 363–379, 1994. 28

[29] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[30] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016.

[31] Weisheng Dong, Lei Zhang, Guangming Shi, and Xiaolin Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *IEEE Trans. Image Processing*, 20(7):1838–1857, 2011.

[32] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[33] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Processing*, 15(12):3736–3745, 2006.

[34] Xueyang Fu, Zheng-Jun Zha, Feng Wu, Xinghao Ding, and John Paisley. Jpeg artifacts reduction via deep convolutional sparse coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2501–2510, 2019.

[35] Huiyi Hu, Brendt Wohlberg, and Rick Chartrand. Task-driven dictionary learning for inpainting. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page nil, 5 2014.

[36] Chen Kong and Simon Lucey. Deep non-rigid structure from motion. *arXiv preprint arXiv:1907.13123*, 2019. 27, 30, 34

[37] Julien Mairal, Francis R. Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[38] Calvin Murdock, Ming-Fang Chang, and Simon Lucey. Deep component analysis via alternating direction neural networks. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, pages 851–867, 2018.

[39] Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, 18:83:1–83:52, 2017.

[40] JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5888–5897, 2017.

[41] J. Sulam, A. Aberdam, A. Beck, and M. Elad. On multi-layer basis pursuit, efficient algorithms and convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.

[42] Jeremias Sulam, Vardan Papyan, Yaniv Romano, and Michael Elad. Multilayer convolutional sparse modeling: Pursuit and dictionary learning. *IEEE Trans. Signal Processing*, 66(15):4090–4104, 2018.

[43] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[44] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 11–20, 2017.

[45] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9446–9454, 2018.

[46] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 350–358, 2012. 28

[47] Jun Xu, Lei Zhang, and David Zhang. A trilateral weighted sparse coding scheme for real-world image denoising. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018. 28

[48] Jun Xu, Lei Zhang, Wangmeng Zuo, David Zhang, and Xiangchu Feng. Patch group based nonlocal self-similarity prior learning for image denoising. In *Proceedings of the IEEE international conference on computer vision*, pages 244–252, 2015.

[49] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

[50] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[51] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time user-guided image colorization with learned deep priors. *CoRR*, abs/1705.02999, 2017.

[52] Bolun Zheng, Rui Sun, Xiang Tian, and Yaowu Chen.

S-net: a scalable convolutional neural network for jpeg compression artifact reduction. *Journal of Electronic Imaging*, 27(4):043037, 2018. 28

[53] Yingying Zhu and Simon Lucey. 27, 30

Convolutional sparse coding for trajectory reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):529–540, 2015. 7, 31, 33

[54] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1–8. IEEE, 2007. 38

[55] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2565–2574, 2020. 38

[56] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *Int. J. Comput. Vis.*, 101(1):6–21, 2013. 38

[57] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Björn Ommer, and Andreas Geiger. Slim: Self-supervised lidar scene flow and motion segmentation. In *Int. Conf. Comput. Vis.*, pages 13126–13136, 2021. 7, 8, 36, 38, 43, 45, 46, 48, 49, 50, 61, 62, 73

[58] Albert E Beaton and John W Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974. 49

[59] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Int. Conf. Comput. Vis.*, pages 7962–7971, 2019. 36

[60] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11621–11631, 2020. 40, 60, 70, 72

[61] Yang Chen and Gerard Medioni. Object modelling by registration of multiple range images. *Img. Vis. Comput.*, 10(3):145–155, 1992. 38, 41, 43, 47, 55, 56

[62] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5939–5948, 2019. 38

[63] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Comput. Vis. Img. Und.*, 89(2-3):114–141, 2003. 38

[64] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d lidar scans. In *Int. Conf. Intel. Rob. Sys.*, pages 1765–1770. IEEE, 2016. 38

[65] Guanting Dong, Yueyi Zhang, Hanlin Li, Xiaoyan Sun, and Zhiwei Xiong. Exploiting rigidity constraints for lidar scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12776–12785, 2022. 38, 46, 48

[66] Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12265–12274, 2020. 38

[67] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Int. Conf. Know. Disc. Data Min.*, KDD'96, page 226–231. AAAI Press, 1996. 46

[68] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981. 46

[69] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3354–3361. IEEE, 2012. 39

[70] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *IEEE Conf. Comput. Vis.*

Pattern Recog., pages 5692–5703, 2021. 4, 7, 8, 36, 37, 38, 43, 46, 47, 48, 49, 50, 55, 56, 60

[71] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3254–3263, 2019. 38

[72] Simon Hadfield and Richard Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *Int. Conf. Comput. Vis.*, pages 2290–2295. IEEE, 2011. 38

[73] Simon Hadfield and Richard Bowden. Scene particles: Unregularized particle-based scene flow estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):564–576, 2013. 38

[74] Michael Himmelsbach, Felix V Hundelshausen, and H-J Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *Intel. Veh. Symp. IV*, pages 560–565. IEEE, 2010. 38

[75] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. Sphereflow: 6 dof scene flow from rgb-d pairs. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3526–3533, 2014. 38

[76] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992. 45

[77] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *User Int. Soft. Tech.*, '11, page 559–568, New York, NY, USA, 2011. Association for Computing Machinery. 38

[78] Víctor Jiménez, Jorge Godoy, Antonio Artuñedo, and Jorge Villagra. Ground segmentation algorithm for sloped terrain and sparse lidar point cloud. *IEEE Access*, 9:132914–132927, 2021. 38

[79] Zhao Jin, Yinjie Lei, Naveed Akhtar, Haifeng Li, and Munawar Hayat. Deformation and correspondence aware unsupervised synthetic-to-real scene flow estimation for point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7233–7243, 2022. 4, 36, 37, 38, 43, 47, 48, 49, 50, 52, 55, 56

[80] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable scene flow from point clouds in the real world. *IEEE Rob. Aut. Letters*, 7(2):1589–1596, 2021. 5, 36, 40, 43, 44

[81] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 46

[82] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[83] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4114–4123, 2021. 4, 37, 38, 47, 48, 49, 52, 55, 56

[84] Seungjae Lee, Hyungtae Lim, and Hyun Myung. Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud. In *Int. Conf. Intel. Rob. Sys.*, pages 13276–13283. IEEE, 2022. 38

[85] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Comput. Graph. For.*, volume 27, pages 1421–1430. Wiley Online Library, 2008. 38

[86] Ruibo Li, Chi Zhang, Guosheng Lin, Zhe Wang, and Chunhua Shen. Rigidflow: Self-supervised scene flow learning on point clouds by local rigidity prior. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 16959–16968, 2022. 38, 42

[87] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Adv. Neural Inform. Process. Syst.*, 34:7838–7851, 2021. 38, 41, 43, 45, 47, 49, 61, 62

[88] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 529–537, 2019. 4, 36, 37, 39, 41, 42, 45

[89] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9246–9255, 2019. 48

[90] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4040–4048, 2016. 36, 37, 39

[91] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Int. Conf. Comput. Vis.*, pages 3061–3070, 2015. 38, 41

[92] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS Annals Photo. Rem. Sens. Spat. Inf. Sci.*, 2:427, 2015. 36, 39

[93] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4460–4470, 2019. 38

[94] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2020. 36, 38, 40

[95] Frank Moosmann, Oliver Pink, and Christoph Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *Intel. Veh. Symp. IV*, pages 215–220. IEEE, 2009. 38

[96] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *Eur. Conf. Comput. Vis.*, pages 424–443. Springer, 2022. 36, 38

[97] Patiphon Narksri, Eijiro Takeuchi, Yoshiki Ninomiya, Yoichi Morales, Naoki Akai, and Nobuo Kawaguchi. A slope-robust cascaded ground segmentation in 3d point cloud for autonomous vehicles. In *Int. Conf. Intel. Trans. Sys.*, pages 497–504. IEEE, 2018. 38

[98] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 165–174, 2019. 38

[99] Mark Pauly, Niloy J Mitra, Joachim Giesen, Markus H Gross, and Leonidas J Guibas. Example-based 3d scan completion. In *Symp. Geom. Proc.*, pages 23–32, 2005. 38

[100] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *Int. J. Comput. Vis.*, 72(2):179–193, 2007. 38

[101] J-P Pons, Renaud Keriven, O Faugeras, and Gerardo Hermosillo. Variational stereovision and 3d scene flow estimation with statistical similarity measures. In *Int. Conf. Comput. Vis.*, volume 2, pages 597–597. IEEE Computer Society, 2003. 38

[102] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *Int. Conf. 3D Vis.*, pages 261–270. IEEE, 2020. 38

[103] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *Eur. Conf. Comput. Vis.*, pages 527–544. Springer, 2020. 48

[104] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017.

[105] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Adv. Neural Inform. Process. Syst.*, 33:7462–7473, 2020. 38

[106] Ivan Tishchenko, Sandro Lombardi, Martin R Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. In *Int. Conf. 3D Vis.*, pages 150–159. IEEE, 2020. 37, 38, 47, 48, 49, 50, 52, 55, 56

[107] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 2, pages 722–729. IEEE, 1999. 36

[108] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Int. Conf. Comput. Vis.*, volume 2, pages 722–729. IEEE, 1999. 38

[109] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023. 43, 49, 70

[110] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a rigid motion prior. In *Int. Conf. Comput. Vis.*, pages 1291–1298. IEEE, 2011. 38

[111] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Int. Conf. Comput. Vis.*, pages 1377–1384, 2013. 38

[112] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Neur. Inform. Process. Syst. Data. Bench. Track*, 2021. 40

[113] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *Eur. Conf. Comput. Vis.*, pages 88–107. Springer, 2020. 4, 36, 37, 38, 47, 48, 49, 52, 55, 56

[114] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. 38

Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *Int. Conf. Rob. Aut.*, pages 5067–5073. IEEE, 2017.

[115] Ramy Battrawy, René Schuster, Mohammad-Ali Nikouei Mahani, and Didier Stricker. RMS-FlowNet: Efficient and Robust Multi-Scale Scene Flow Estimation for Large-Scale Point Clouds. In *Int. Conf. Rob. Aut.*, pp. 883–889. IEEE, 2022.

[116] Michael Black. Novelty in science: A guide to reviewers. https://medium.com/@black_51980/novelty-in-science-8f1fd1a0a143, 2022.

[117] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, 2021. 56

[118] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

[119] Nathaniel Chodosh, Deva Ramanan, and Simon Lucey. Re-Evaluating LiDAR Scene Flow for Autonomous Driving. *arXiv preprint*, 2023. 8, 54, 55, 56, 59

[120] Emeç Erçelik, Ekim Yurtsever, Mingyu Liu, Zhijie Yang, Hanzhen Zhang, Pınar Topçam, Maximilian Listl, Yılmaz Kaan Çaylı, and Alois Knoll. 3D Object Detection with a Self-supervised Lidar Scene Flow Backbone. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision – ECCV 2022*, pp. 247–265, Cham, 2022. Springer Nature Switzerland.

[121] Shengyu Huang, Zan Gojcic, Jiahui Huang, and Konrad Schindler Andreas Wieser. Dynamic 3D Scene Analysis by Point Cloud Accumulation. In *European Conference on Computer Vision, ECCV*, 2022.

[122] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable Scene Flow From Point Clouds in the Real World. *IEEE Robotics and Automation Letters*, 12 2021. 52, 53, 54, 55, 56, 58

[123] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything. *arXiv:2304.02643*, 2023.

[124] Alex Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12689–12697, 2019. 52, 54

[125] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 473–488. Springer, 2020. 8, 56

[126] Ruibo Li, Guosheng Lin, Tong He, Fayao Liu, and Chunhua Shen. HCRF-Flow: Scene flow from point clouds with continuous high-order CRFs and position-aware flow embedding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 364–373, 2021a.

[127] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural Scene Flow Prior. *Advances in Neural Information Processing Systems*, 34, 2021b. 52, 53, 54, 55, 56

[128] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning Scene Flow in 3D Point Clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 52

[129] Nikola Lopac, Irena Jurdana, Adrian Brnelić, and Tomislav Krljan. Application of Laser Systems for Detection and Ranging in the Modern Road Transportation and Maritime Sector. *Sensors*, 22(16), 2022. ISSN 1424-8220.

[130] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training. *arXiv preprint arXiv:2210.00030*, 2022.

[131] Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-Image Representations and Rewards for Robotic Control. *arXiv preprint arXiv:2306.00958*, 2023.

[132] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[133] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R. Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion Inspired Unsupervised Perception and Prediction in Autonomous Driving. European Conference on Computer Vision (ECCV), 2022. 59

[134] OpenAI. Gpt-4 technical report, 2023. 56

[135] Neehar Peri, Mengtian Li, Benjamin Wilson, Yu-Xiong Wang, James Hays, and Deva Ramanan. An empirical analysis of range for 3d object detection. *arXiv preprint arXiv:2308.04054*, 2023. 8, 55

[136] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[137] Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A Universal Visual Representation for Robot Manipulation. *Conference on Robot Learning (CoRL) 2022*, 03 2022.

[138] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 52, 54

[139] Kyle Vedder and Eric Eaton. Sparse PointPillars: Maintaining and Exploiting Input Sparsity to Improve Runtime on Embedded Systems. *International Conference on Intelligent Robots and Systems (IROS)*, 2022. 54

[140] *Velodyne Lidar Alpha Prime*. Velodyne Lidar, 11 2019. 52

[141] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv: Arxiv-2305.16291*, 2023.

[142] Jun Wang, Xiaolong Li, Alan Sullivan, Lynn Abbott, and Siheng Chen. PointMotionNet: Point-Wise Motion Learning for Large-Scale LiDAR Point Clouds Sequences. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 4418–4427, 2022.

[143] Xinshuo Weng, Jianren Wang, Sergey Levine, Kris Kitani, and Nicholas Rhinehart. Inverting the pose forecasting pipeline with spf2: Sequential pointcloud forecasting for sequential pose forecasting. In *Conference on robot learning*, pp. 11–20. PMLR, 2021.

[144] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021. 52, 54

[145] Bo Yang, Ishan Khatri, Michael Happold, and Chulong Chen. Adcnet: Learning from raw radar data via distillation, 2023.

[146] Guangyao Zhai, Xin Kong, Jinhao Cui, Yong Liu, and Zhen Yang. FlowMOT: 3D Multi-Object Tracking by Scene Flow Association. *ArXiv*, abs/2012.07541, 2020.

[147] Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. PointOdyssey: A Large-Scale Synthetic Dataset for Long-Term Point Tracking. In *ICCV*, 2023.

[148] Bibby, C., Reid, I.: Simultaneous localisation and mapping in dynamic environments (slamide) with reversible data association. In: Proceedings of Robotics: Science and Systems. vol. 66, p. 81 (2007) 63

[149] Chodosh, N., Simon, L., Deva, R.: Re-evaluating lidar scene flow (2024) 60, 61, 62

[150] Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised nerf: Fewer views and faster training for free. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12882–12891 (2022) 71, 72

[151] Dewan, A., Caselitz, T., Tipaldi, G.D., Burgard, W.: Motion-based detection and tracking in 3d lidar scans. In: 2016 IEEE international conference on robotics and automation (ICRA). pp. 4508–4513. IEEE (2016) 63

[152] Hähnel, D., Schulz, D., Burgard, W.: Map building with mobile robots in populated environments. In: IROS. pp. 496–501 (2002) 63

[153] Henein, M., Zhang, J., Mahony, R., Ila, V.: Dynamic slam: The need for speed. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 2123–2129. IEEE (2020) 63

[154] Huang, J., Gojcic, Z., Atzmon, M., Litany, O., Fidler, S., Williams, F.: Neural kernel surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4369–4379 (2023) 67, 71, 72

[155] Jiang, C., Fougerolle, Y., Fofi, D., Demonceaux, C.: Dynamic 3d scene reconstruction and enhancement. In: Image Analysis and Processing-ICIAP 2017: 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part I 19. pp. 518–529. Springer (2017) 63

[156] Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7, p. 0 (2006) 67

[157] Li, H., Yu, J., Ye, Y., Bregler, C.: Realtime facial animation with on-the-fly correctives. ACM Trans. Graph. **32**(4), 42–1 (2013) 63

[158] Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., Yang, B., Ma, W.C., Urtasun, R.: Lidarsim: Realistic lidar simulation by leveraging the real world. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11167–11176 (2020) 63

[159] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021) 72

[160] Mitra, N.J., Flöry, S., Ovsjanikov, M., Gelfand, N., Guibas, L.J., Pottmann, H.: Dynamic geometry registration. In: Symposium on geometry processing. pp. 173–182 (2007) 63

[161] Newcombe, R.A., Fox, D., Seitz, S.M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 343–352 (2015) 60, 63

[162] Palafox, P., Bozic, A., Thies, J., Niessner, M., Dai, A.: Npms: Neural parametric models for 3d deformable shapes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12695–12705 (2021) 63

[163] Peri, N., Dave, A., Ramanan, D., Kong, S.: Towards long-tailed 3d detection. In: Conference on Robot Learning. pp. 1904–1915. PMLR (2023) 70, 71, 72

[164] Qian, C., Sun, X., Wei, Y., Tang, X., Sun, J.: Realtime and robust hand tracking from depth. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1106–1113 (2014) 63

[165] Runz, M., Buffier, M., Agapito, L.: Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In: 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 10–20. IEEE (2018) 60

[166] Taylor, J., Shotton, J., Sharp, T., Fitzgibbon, A.: The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 103–110. IEEE (2012) 63

[167] Tian, R., Zhang, Y., Cao, Z., Zhang, J., Yang, L., Coleman, S., Kerr, D., Li, K.: Object slam with robust quadric initialization and mapping for dynamic outdoors. IEEE Transactions on Intelligent Transportation Systems (2023) 63

[168] Turki, H., Zhang, J.Y., Ferroni, F., Ramanan, D.: Suds: Scalable urban dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12375–12385 (2023) 71, 72

[169] Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant cnns. In: 2017 international conference on 3D Vision (3DV). pp. 11–20. IEEE (2017) 61

[170] Vizzo, I., Chen, X., Chebrolu, N., Behley, J., Stachniss, C.: Poisson surface reconstruction for lidar odometry and mapping. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 5624–5630. IEEE (2021) 63

[171] Wang, C.C., Thorpe, C., Thrun, S., Hebert, M., Durrant-Whyte, H.: Simultaneous localization, mapping and moving object tracking. The International Journal of Robotics Research **26**(9), 889–916 (2007) 63

[172] Wang, J., Manivasagam, S., Chen, Y., Yang, Z., Bârsan, I.A., Yang, A.J., Ma, W.C., Urtasun, R.: Cadsim: Robust and scalable in-the-wild 3d reconstruction for controllable sensor simulation. In: Conference on Robot Learning. pp. 630–642. PMLR (2023) 63

[173] Wang, Y., Dai, Y., Liu, Q., Yang, P., Sun, J., Li, B.: Cu-net: Lidar depth-only completion with coupled u-net. IEEE Robotics and Automation Letters **7**(4), 11476–11483 (2022) 61

[174] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., Ramanan, D., Carr, P., Hays, J.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021) (2021) 60, 70

[175] Yang, G., Vo, M., Neverova, N., Ramanan, D., Vedaldi, A., Joo, H.: Banmo: Building animatable 3d neural models from many casual videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2863–2873 (2022) 60

[176] Yang, S., Scherer, S.: Cubeslam: Monocular 3-d object slam. IEEE Transactions on Robotics **35**(4), 925–938 (2019) 63

[177] Yang, Z., Manivasagam, S., Chen, Y., Wang, J., Hu, R., Urtasun, R.: Reconstructing objects in-the-wild for realistic sensor simulation 63

[178] Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11784–11793 (2021) 71

[179] Yu, T., Zheng, Z., Guo, K., Liu, P., Dai, Q., Liu, Y.: Function4d: Real-time human volumetric capture from very sparse consumer rgbd sensors. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5746–5756 (2021) 60

[180] Zhao, Y., Bai, L., Zhang, Z., Huang, X.: A surface geometry model for lidar depth completion. IEEE Robotics and Automation Letters **6**(3), 4457–4464 (2021) 61

[181] Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018) 71