This thesis is presented for the degree of Doctor of Philosophy of The University of Adelaide

Deeply Learned Priors for Geometric Reconstruction

THE UNIVERSITY *of* ADELAIDE

SUB CRUCE LUMEN

Chamara Saroj Weerasekera

November 2018

School of Computer Science

Supervisors:
Prof. Ian Reid
Dr. Ravi Garg

Thesis Declaration

Signature:

Date: 17/8/2018

Abstract

This thesis comprises of a body of work that investigates the use of deeply learned priors for dense geometric reconstruction of scenes. A typical image captured by a 2D camera sensor is a lossy two-dimensional (2D) projection of our three-dimensional (3D) world. Geometric reconstruction approaches usually recreate the lost structural information by taking in multiple images observing a scene from different views and solving a problem known as Structure from Motion (SfM) or Simultaneous Localization and Mapping (SLAM). Remarkably, by establishing correspondences across images and use of geometric models, these methods (under reasonable conditions) can reconstruct a scene's 3D structure as well as precisely localise the observed views relative to the scene. The success of *dense* every-pixel multi-view reconstruction is however limited by matching ambiguities that commonly arise due to uniform texture, occlusion, and appearance distortion, among several other factors.

The standard approach to deal with matching ambiguities is to handcraft priors based on assumptions like piecewise smoothness or planarity in the 3D map, in order to "fill in" map regions supported by little or ambiguous matching evidence. In this thesis we propose *learned* priors that in comparison more closely model the true structure of the scene and are based on geometric information *predicted* from the images. The motivation stems from recent advancements in deep learning algorithms and availability of massive datasets, that have allowed Convolutional Neural Networks (CNNs) to predict geometric properties of a scene such as point-wise surface normals and depths, from just a single image, more reliably than what was possible using previous machine learning-based or hand-crafted methods.

In particular, we first explore how single image-based surface normals from a CNN trained on massive amount of indoor data can benefit the accuracy of dense reconstruction given input images from a moving monocular camera. Here we propose a novel surface normal based inverse depth regularizer and compare its performance against the inverse depth smoothness prior that is typically used to regularize regions in the reconstruction that are textureless. We also propose the first real-time CNN-based framework for live dense monocular reconstruction using our learned normal prior.

Next, we look at how we can use deep learning to learn features in order to improve the pixel matching process itself, which is at the heart of multi-view geometric reconstruction. We propose a self-supervised feature learning scheme using RGB-D data from a 3D sensor (that does not require any manual labelling) and a multi-scale CNN architecture for feature extraction that is fast and efficient to run inside our proposed real-time monocular reconstruction framework. We extensively analyze the combined benefits of using learned normals and deep features that are good-for-matching in the context of dense reconstruction, both quantitatively and qualitatively on large real world datasets.

Lastly, we explore how learned depths, also predicted on a per-pixel basis from a single image using a CNN, can be used to inpaint sparse 3D maps obtained from monocular SLAM or a 3D sensor. We propose a novel model that uses predicted depths and confidences from CNNs as priors to inpaint maps with arbitrary scale and sparsity. We obtain more reliable reconstructions than those of traditional depth inpainting methods such as the cross-bilateral filter that in comparison offer few learnable parameters. Here we advocate the idea of "just-in-time reconstruction" where a higher level of scene understanding reliably inpaints the corresponding portion of a sparse map on-demand and in real-time.

Acknowledgments

I am first and foremost very grateful to my principal supervisor, Prof. Ian Reid, for inspiring this work, and for giving me invaluable guidance throughout the time of my PhD, which made this work possible, and has helped and inspired me to grow as a researcher. I greatly value the opportunity that was given to me to be a part of a great research team.

My sincere thanks also goes to my co-supervisor, Dr. Ravi Garg, for the immense guidance and support, and the countless fruitful discussions leading to the ideas in this work.

I am also very thankful to Dr. Yasir Latif for the invaluable discussions, introduction into the world of SLAM, and great support towards this work.

I also like to thank my former co-supervisor, Prof. Chunhua Shen, for providing me with valuable insights and advice, Dr. Guosheng Lin, Dr. Cesar Cadena and Dr. Trung Pham for the great explanations, support and advice, Thanuja Dharmasiri and Prof. Tom Drummond for their great support and the rewarding research collaboration, Dr. Nick Carlevaris for the resourceful knowledge to help me get started, Dr. Niko Sünderhauf for the useful remote meetings, and Dr. Alireza Khosravian for the helpful advice.

I wish to thank all my lab mates, both former and present, for the stimulating discussions, the support, and the lasting memories. I especially like to thank Huangying Zhang and Mehdi Hosseinzadeh for the great help and discussions towards this work.

I also like to acknowledge Prof Victor Prisacariu for the memorable opportunity to visit the Active Vision Lab and its research group at the University of Oxford.

I am grateful for the support and memories given by the past and current members of the School of Computer Science at the University of Adelaide, the Australian Centre for Visual Technologies (Australian Institute for Machine Learning), and the Australian Centre for Robotic Vision, and last but not least, to my parents and my brother for their timeless support, and always being there to share the experiences from home and my PhD journey.

Dedicated to —
— my parents.

Publications

This thesis contains the following work that has been published or prepared for publication:

- **C. S. Weerasekera**, Y. Latif, R. Garg, I. Reid, "Dense Monocular Reconstruction using Surface Normals", *ICRA 2017*
- **C. S. Weerasekera**, R. Garg, Y. Latif, I. Reid, "Learning Deeply Supervised Good Features to Match for Dense Monocular Reconstruction", *ACCV 2018*
- H. Zhan, R. Garg, **C. S. Weerasekera**, K. Li, H. Agarwal, I. Reid, "Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction", *CVPR 2018*
- **C. S. Weerasekera**, T. Dharmasiri, R. Garg, T. Drummond, I. Reid, "Just-in-Time Reconstruction: Inpainting Sparse Maps using Single View Depth Predictors as Priors", *ICRA 2018*

Contents

Contents vii

List of Figures xi

List of Tables xv

1 Introduction 1

1.1 Motivation 1

1.2 Background 4

1.2.1 Multi-View Geometric Reconstruction 4

1.2.2 Scene Understanding for Geometry Estimation 9

1.3 Contributions 14

1.4 Thesis Outline 17

Bibliography 19

2 Literature Review 21

2.1 Multi-View Geometric Reconstruction 21

2.1.1 Offline Structure from Motion (SfM) 21

2.1.2 Visual Simultaneous Localization and Mapping (SLAM) 22

2.1.3 Online Sparse Methods 26

2.1.4 Online Dense Methods 28

2.1.5 Image Correspondence Estimation 31

2.2 Scene Understanding for Geometry Estimation 34

2.2.1 Convolutional Neural Networks (CNNs) 34

2.2.2 Convolutional Neural Networks for Geometric Scene Understanding 37

2.2.3 Conditional Random Fields (CRFs) 42

2.3 Legendre-Fenchel Transform 44

2.4 Primal-Dual Optimisation 46

2.5 Summary 51

Bibliography 53

3 Dense Monocular Reconstruction using Surface Normals **63**
3.1 Introduction . . . 63
3.2 Background and Related Work . . . 65
3.3 Method . . . 66
3.3.1 Notations . . . 66
3.3.2 Energy Formulation for Keyframe Reconstruction . . . 66
3.3.3 Optimisation of Keyframe Inverse Depths . . . 70
3.3.4 Camera Tracking and Frame selection . . . 71
3.3.5 Scaling Camera Translations . . . 72
3.3.6 Surface Normals Prediction . . . 72
3.3.7 Volumetric Fusion . . . 74
3.4 Experimental Results . . . 74
3.5 Future Directions . . . 80
3.5.1 Outdoor Experiments . . . 80
3.5.2 Learning Surface Normal Prior Confidences . . . 82
3.5.3 Implicit Learning of Surface Normals . . . 83
3.5.4 More Efficient Surface Normal Estimation . . . 84
3.6 Conclusion . . . 88

Bibliography **91**

4 Learning Deeply Supervised Good Features to Match for Dense Monocular Reconstruction **97**
4.1 Introduction . . . 97
4.2 Background and Related Work . . . 99
4.3 Method . . . 101
4.3.1 Neural Network Architecture . . . 101
4.3.2 Training . . . 103
4.3.3 Keyframe Reconstruction Using Deep Features . . . 105
4.4 Experimental Results . . . 106
4.4.1 Evaluation of Learned Features for Matching . . . 106
4.4.2 Learned Features as Feature Reconstruction Loss for Unsupervised Single View Depth Estimation . . . 111
4.5 Future Directions . . . 113
4.5.1 Learning Feature Matching Confidences . . . 113
4.5.2 A Convex Approximation to the Multi-View Observation Model for Keyframe Reconstruction . . . 114
4.5.3 Introducing Additional Features for Matching . . . 115
4.5.4 Learned Features for Dense Alignment-Based Camera Tracking . . . 115
4.5.5 Learned Features for Other Computer Vision Tasks . . . 117
4.6 Conclusion . . . 117

Bibliography **135**

5 Just-in-Time Reconstruction: Inpainting Sparse Maps using Single View Depth Predictors as Priors **139**

5.1 Introduction . . . 139

5.2 Just-in-Time Reconstruction . . . 141

5.2.1 Model Definition . . . 142

5.2.2 Inference Method . . . 144

5.2.3 Learning to Predict Confidence Weights . . . 147

5.3 Experimental Results . . . 149

5.3.1 Inpainting Sparse Depth Maps from Multi-View Geometry . . . 149

5.3.2 Inpainting Sparse Depth Maps from Depth Sensor . . . 153

5.3.3 Runtimes . . . 156

5.4 Future Directions . . . 156

5.4.1 Increasing Model Expressibility . . . 156

5.4.2 An Inverse Depth Parameterization . . . 159

5.4.3 Learning Scale-Invariant Depth Prior Confidences . . . 160

5.4.4 A Unified Model for Just-In-Time Reconstruction . . . 165

5.5 Conclusion . . . 168

Bibliography **169**

6 Conclusion and Future Directions **173**

6.1 Conclusion . . . 173

6.2 Future Directions . . . 174

Bibliography **179**

List of Figures

1.1 The figure illustrates the abstract notion of a *cost volume*, and the matching ambiguity that arise during dense reconstruction. 3

1.2 Figure illustrates point correspondence geometry. 5

1.3 The figure illustrates more in depth detail about the matching problem in texture-less regions and how the surface normal prior that we propose in Chapter 3 helps guide the solution towards the ideal minimum, in contrast to the weaker smoothness prior . . . 8

1.4 The figure illustrates how the learned features that are good for matching which we propose in Chapter 4 enable more reliable matching even for low-image gradient pixels, and make the matching error more convex. 9

1.5 The figure illustrates how both our learned surface normal prior (Chapter 3) and learned features that are good for matching (Chapter 4), improve the quality of dense reconstruction. 10

1.6 Conceptualization of "Just-in-Time Reconstruction". 11

1.7 The figure illustrates results of our proposed "just-in-time" reconstruction framework in Chapter 5. 12

1.8 Multi-scale CNN architecture for regressing a depth map, normal map, and semantic label map from a single image. 14

2.1 A high level abstraction of the higher modeling complexity of filter-based methods in comparison to keyframe based methods for the same number of landmarks. 27

2.2 Illustration of the ORB-SLAM pipeline. 28

2.3 Illustration of some of the most popular visual SLAM systems over the years. 29

2.4 The Truncated Sign Distance Function (TSDF) for representing volumetric reconstructions. 31

2.5 The architecture LeNet-5, of one of the successful pioneering CNNs applied for hand written character recognition. 35

2.6 Plots for the negative likelihood of four instances of the generalised Gaussian distribution 43

2.7 Illustration of the Legrendre Fenchel Transform. 45

2.8 Figure depicts some examples of the Legrendre-Fenchel Transform for L1 norm, squared L2 norm, and Huber norm respectively. 47

3.1 Reconstruction using smoothness regularizer and normal prior. 64

3.2 A neighbouring 3D point pair and corresponding surface normal. 67

3.3 Key components and data flows of our framework for real-time live dense monocular reconstruction. 73
3.4 Average keyframe RMS reconstruction error (m) w.r.t regularization strength on the raw NYU-D V2 test sequences. 75
3.5 Qualitative results on NYU raw dataset 'bathroom_0003' test sequence. 76
3.6 Qualitative results on NYU raw dataset 'bedroom_0048' test sequence. 77
3.7 Qualitative results on TUM dataset 'fr2_desk' sequence. 78
3.8 Dense volumetric reconstruction results of fusing single view depth predictions, and comparison with other methods. 80
3.9 The figure illustrates the complementary information present in both depth and normal predictions. 81
3.10 Our shallow CNN architecture for fast but less accurate surface normal prediction. . . 86
3.11 Comparison of different surface normal prediction methods against our proposed fast architecture. 87

4.1 Replacing color features with learned deep features and hand crafted priors with learned priors improves the reconstruction quality. 98
4.2 Overview of the proposed multi-scale network architecture for regressing good features to match, and the training setup. 102
4.3 The effect of changing regularization strength on reconstruction accuracy. 107
4.4 Generalizability of learned features trained on NYU dataset to a completely different type of scene on the KITTI stereo dataset without any finetuning. 108
4.5 A snapshot of our results on the 'living_room_0075' sequence from the NYU-D V2 raw dataset. 119
4.6 Qualitative comparison of reconstructions obtained by our mono SLAM framework for the sequence 'living_room_0080' against those of baselines. 120
4.7 Qualitative comparison of reconstructions obtained by our mono SLAM framework for the sequence 'kitchen_0046' against those of DTAM. 121
4.8 Comparison of keyframe reconstructions (based purely on matching with 1 frame) using our learned features against those of baselines (example set 1) 122
4.9 Comparison of keyframe reconstructions (based purely on matching with 30 frames) using our learned features against those of baselines (example set 1) 123
4.10 Comparison of keyframe reconstructions (based purely on matching with 1 frame) using our learned features against those of baselines (example set 2) 124
4.11 Comparison of keyframe reconstructions (based purely on matching with 30 frames) using our learned features against those of baselines (example set 2) 125
4.12 Comparison of keyframe reconstructions (based purely on matching with 1 frame) using our learned features against those of baselines (example set 3) 126
4.13 Comparison of keyframe reconstructions (based purely on matching with 30 frames) using our learned features against those of baselines (example set 3) 127

4.14 Figure compares RGB and learned features for keyframe reconstruction purely based on feature matching for: a bathroom, a dining room, and a bedroom sequence from the NYU-D v2 dataset. 128
4.15 Figure shows the cost-volume as it evolves after accumulating evidences from multiple frames for 3 different points, using learned features and rgb features. 129
4.16 Visual ablation study on the effect of using different feature types for stereo matching in the KITTI dataset. 130
4.17 Comparison of the performance of low-level features trained for image classification, for pixel matching, against that of our learned features. 131
4.18 Output of the fast version of our learned matching feature network 132
4.19 Output of the full version of our learned matching feature network 133

5.1 Three possible scenarios where our "just-in-time reconstruction" method for high-level image-guided inpainting of sparse depth maps may be applied. 140
5.2 Qualitative ablation of the impact of local "data-driven smoothing" on the reconstruction as enforced by our local CRF pairwise terms, and the impact of incorporating confidences into the energy. 145
5.3 Figure shows the result of densification of a sparse map via Delaunay triangulation, for use as input to our confidence prediction CNN, and the final point-wise confidence prediction for the sparse map. 148
5.4 Plots showing the average scale invariant error of our just-in-time reconstructions for sparse ORB-SLAM map input on the NYU dataset as we vary the hyperparameters. . 151
5.5 ORB-SLAM inpainting results for some of the images in the NYU test set. 151
5.6 Comparison of our method against baselines for inpainting ORB-SLAM depth maps in the NYU dataset. 152
5.7 Comparison of our method against baselines for inpainting Kinect depth maps with randomly removed data in the NYU dataset. 153
5.8 Intermediate and final outputs for LIDAR depth map inpainting in the KITTI dataset. 154
5.9 Comparison of our method against baselines for inpainting LIDAR depth maps in the KITTI dataset. 154

6.1 This figure shows the results of naively fusing (voxel color averaging) semantic label maps predicted from each keyframe into a voxel grid constructed using the keyframe depth maps estimated by our framework and poses given by monocular ORB-SLAM tracking. 175

List of Tables

3.1 Quantitative results on 25 raw NYU-D V2 Dataset test sequences. 75

3.2 Quantitative results on the TUM dataset 'fr2_desk' sequence. 76

3.3 Quantitative results on ICL-NUIM dataset 'lr kt0' sequence. 76

3.4 The table demonstrates the need for normalization of the difference vectors in 3D, during implicit learning of normals 85

3.5 Surface normals prediction accuracy measured in degrees against the ground truth. . . 88

4.1 Quantitative results on 25 raw NYU-D V2 Dataset test sequences. 109

4.2 Quantitative results on the TUM dataset 'fr2_desk' sequence. 109

4.3 Quantitative results on ICL-NUIM dataset 'lr kt0' sequence. 109

4.4 Ablation study on the effect of using different feature types and stereo/monocular sequences during training for single view depth estimation in the KITTI dataset. . . . 112

5.1 Quantitative results of inpainting ORB-SLAM maps on NYUv2 dataset, followed by an ablation of the effects of incorporating learned pointwise confidences of the sparse map and predicted depth map. 150

5.2 Inpainting results on the NYU dataset for Kinect depth maps which are further sparsified by removing random crops. 155

5.3 Inpainting results on the KITTI dataset for LIDAR depth maps which are further sparsified by randomly removing a 2/3rd of its points. 155

Chapter 1

Introduction

In this chapter we motivate the work in this thesis and provide a brief background and insight into the underlying problem. We then summarize our key contributions, and provide an outline of the rest of the thesis.

1.1 Motivation

The field of computer vision aims to enable one of the major features of an intelligent system — *the ability to capture, process, and interpret visual information.* Among the various focus areas in this field, two of the most prevalent are geometric reconstruction and scene understanding. While scene understanding is focused on extracting high level semantics (e.g. objects, people, actions, etc.) from one or more images, geometric reconstruction aims to re-create the three dimensional (3D) structure of environments that seemingly vanish after being projected onto two-dimensional (2D) planes, i.e. the images. One of the ultimate goals common to both areas is to create systems that can generate some form of detailed and semantically meaningful 3D representation of the environment from just raw images, analogous to how humans “see” and perceive the world. Naturally, such computer vision-based systems have potential uses in diverse applications, and are a driving-force behind new and improved assistive and fully-autonomous technologies. Some notable applications include autonomous (or assistive) navigation, autonomous object manipulation and other forms of environmental interaction, automatic detailed visual analysis of unsafe or unreachable environments (from inside a human body to outer space), surveillance, augmented reality, and many others.

The area of geometric reconstruction typically involves use of multi-view geometry methods. They exploit the remarkable fact that, under reasonable assumptions, images captured from multiple views (such as those captured by a standard monocular camera moving in an environment), provide sufficient photometric and geometric information to reconstruct the 3D structure of the observed scene, while also precisely localising the camera’s position and orientation within that scene. Multi-view geometric reconstruction however is heavily reliant on the correctness of per-pixel correspondence estimation between images, and matching ambiguities are the main source of error for *dense* reconstruction (Figure 1.1). The standard approach to resolve these ambiguities is to handcraft priors based on some (often non-ideal) assumption about the scene structure, such as piecewise smoothness, in order to propagate the more reliable evidence and *regularize* or “fill-in” parts of the reconstruction that otherwise will end up being incorrectly reconstructed. Another common and complementary approach is to handcraft better features that are *good for matching*. However it is a difficult task to handcraft optimal features for *dense* correspondence between images, especially given there can be limited/repetitive texture in the images, poor photometric consistency due to lighting changes, or severe appearance distortion when the images are captured at arbitrary viewpoints with large relative rotation and translation. Most multi-view reconstruction methods resort to performing sparse or semi-dense reconstruction due to this reason (scalability being another factor).

The area of scene understanding, on the other hand, involves methods for extracting some desired high-level information about the observed scene and its contents (objects, persons, actions, etc.), typically based on suitable visual features derived from one or more images, and use of a *machine learning* algorithm to map these features onto the desired output. Key problems in this area include detection, classification, and segmentation, of the desired entities in the images. Of more interest to this work is scene understanding algorithms being applied to extract geometric attributes about a scene and its contents (e.g. per-pixel surface orientations, and even per-pixel depth). These geometric attributes, while often unused, present themselves as a rich source of information to complement multi-view geometric reconstruction. In fact, the human ability of understanding the geometry of a scene even with one eye closed (and without moving), albeit with reduced precision, provides inspiration that rich geometric information can be learned to be inferred from just a single image. Traditional scene understanding approaches relied upon handcrafted features to suit each task. However, recent developments in deep learning methods and biologically inspired Convolutional Neural Networks (CNNs), on top of availability of large datasets and computational resources, have enabled automatic and end-to-end learning of features and mappings from raw input to output, outperforming traditional methods in almost all tasks in (and out of) computer vision. Of particular interest to our work is the relatively reliable prediction of per-point surface normals and depth of a scene from just a single image, via CNNs.

In this thesis, we propose *how* this deeply learned geometric information via CNNs can be put to use in order to complement traditional *dense* geometric reconstruction approaches, and explore *when* such learned information is most useful based on extensive evaluation on real world data. We also propose a novel way of using a CNN to automatically learn features that are good for matching, in order to address the ambiguities that arise in dense pixel matching — the root issue that plagues dense multi-view geometric reconstruction. Our main idea throughout is to exploit the *automatic* and *end-to-end* learning capabilities of CNNs for geometric reconstruction (so as to limit the amount of handcrafting otherwise needed), in combination with existing well-established geometric methods. The following sections provide a brief background on geometric reconstruction, the problem of per-pixel scene depth estimation by exploiting multi-view point correspondence geometry, how matching ambiguities are typically addressed, and CNNs for geometry estimation. Most of these concepts are revisited in more detail in Chapter 2. At various points of this introduction we also discuss where and how our contributions fit in.

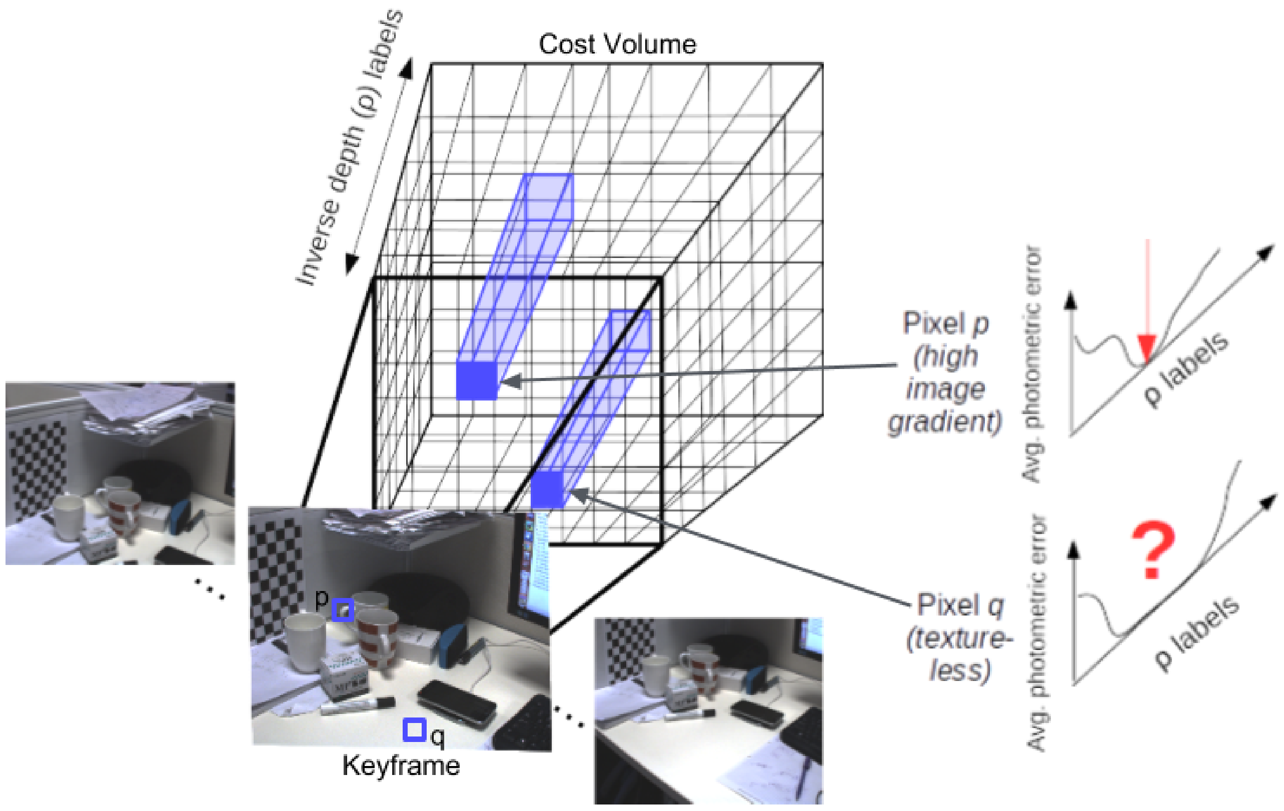


Figure 1.1: The figure illustrates the abstract notion of a *cost volume* as used in DTAM (Newcombe et al. [2011]), and the matching ambiguity that arise during dense reconstruction. Each entry in the cost volume stores the average matching error (based on a series of overlapping images) associated for a particular pixel in the *keyframe* image, for a particular discrete depth (or inverse depth) hypothesis. Through brute force search we can try different possible depth hypotheses for each pixel in the keyframe, to find the depth label that gives the lowest average matching error. The matching process happens along unique 2D lines (epipolar lines) in each of the overlapping live images (see Figures 1.2 and 1.3). Each epipolar line in the live image is defined by the camera transformation (rotation and translation) between the keyframe and the respective live image, as well as the location of the pixel in the keyframe that we want to match. In this example point p is easily matched to other images as it has high image gradient, usually corresponding to a distinct minimum in the average (matching) photometric error, in which case its depth can be inferred directly from the cost volume. Point q is difficult to match to other images as there is little texture around it, and this necessitates the need for good scene priors to guide the solution (Chapters 3 and 5), and/or better features that are suited for matching (Chapter 4).

1.2 Background

1.2.1 Multi-View Geometric Reconstruction

Perceiving the 3D structure of the environment is a trivial task for humans with our highly developed vision system. This task however is incredibly demanding for computers given just images which in their raw form are merely numbers representing color of what's projected from the 3D world. The area of geometric reconstruction aims to develop algorithms in order to solve the problem of recovering the lost 3D structure from what is projected onto images, typically relying on two or more images observing that scene from different views as input. Many applications directly desire detailed 3D knowledge (e.g. autonomous navigation, object manipulation, detailed environment/structural inspection, 3D model generation for computer aided design, augmented reality, etc.), and for those applications that also desire more high-level scene knowledge, the added insight about scene geometry benefit general scene understanding as well. A 2D camera is a bearing-only sensor that provides direction, but not the distance, from the camera center to each 3D landmark (point of interest in the scene). Therefore, multiple image readings must be acquired from at least two views with some baseline (relative *translation* between the views) in order to have sufficient parallax to "triangulate" the location of each desired landmark (Figure 1.2). This means that the camera must move in the case of a monocular setup (more challenging as scene characteristics can change over time), but movement is optional in the case of a stereo or multi-camera setup.

To recover the 3D position of a landmark, an implicit or explicit data-association step is necessary to match the observations in two or more images belonging to that landmark. If the pose between a pair of images in unknown, we can in many cases recover it simply by establishing correspondences between the two images for a *set* of landmarks (that are part of the rigid portion of the scene) and satisfying epipolar geometry constraints (Hartley and Zisserman [2003]; Longuet-Higgins [1981]; Nister [2003]). This is one of the many remarkable feats in geometric reconstruction as relative pose is recovered based on landmark bearing information alone, allowing the reconstruction process to be "bootstraped". In comparison it becomes an easier task to estimate relative pose when given landmark range information as well, and vice versa. The problem of recovering both the 3D landmark locations and poses of the images (or the camera trajectory in the case of a moving camera) is referred to as Structure from Motion (SfM) or visual Simultaneous Localisation and Mapping (SLAM).

SfM methods mostly deal with an unordered set of input images, typically processing the images in an offline manner as a batch, and performing global bundle adjustment (adjusting for all landmark locations, camera poses and camera intrinsics such that they agree with all the visual observations). This is in contrast to visual SLAM methods that aim to process images as and when they are captured by a moving camera, in an incremental manner and in real-time. Modern SLAM systems track the camera at frame rate, perform mapping at near-frame rate, with local and global bundle-adjustment running optionally in the back-end. SLAM methods are therefore of particular importance to autonomous robots that are almost always required to know, at any given point in time, the best possible estimate of the 3D map of its unknown environment, as well as their precise location within it, by making most use of the information available up until any given moment.

Geometric reconstruction methods in general most often make some form of reasonable assumption

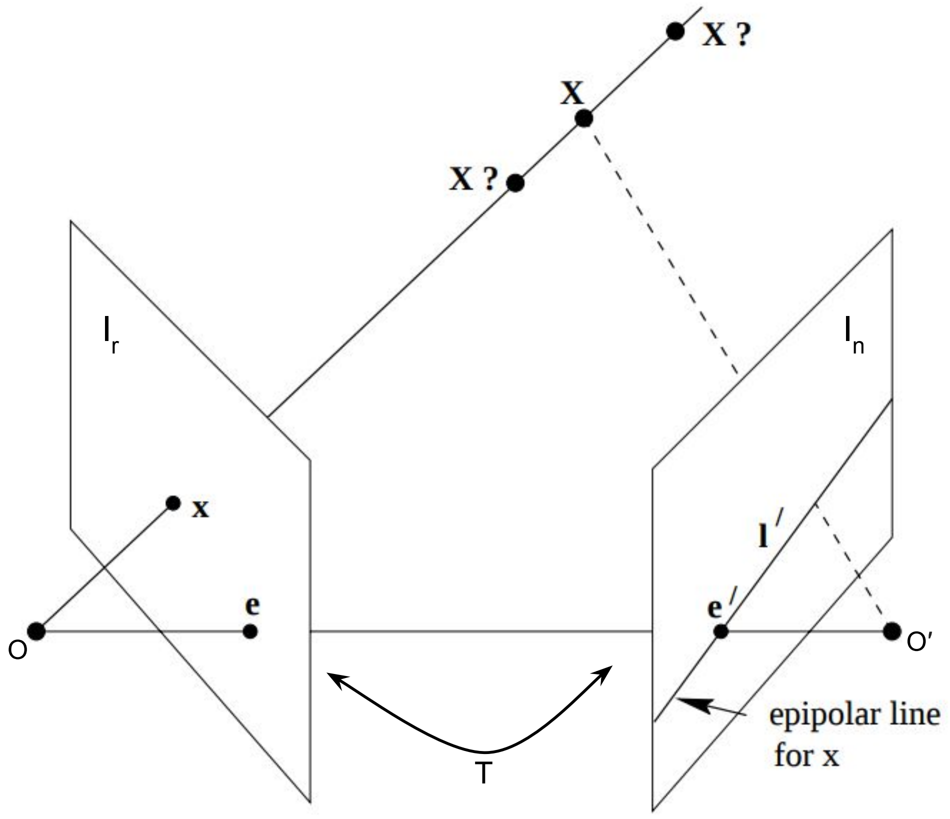


Figure 1.2: Figure illustrates the point correspondence geometry involved when reconstructing (or "triangulating") a 3D point $\mathbf{X}$ for pixel $\mathbf{x}$ in the keyframe I_r. The image point $\mathbf{x}$ back-projects to a ray in 3D defined by I_r's camera centre, O, and $\mathbf{x}$. This ray is imaged as a line $l^{'}$ in the live image I_n, i.e. the epipolar line for $\mathbf{x}$. The 3D point $\mathbf{X}$ which projects to $\mathbf{x}$ must lie on this ray, so the image of $\mathbf{X}$ in the second view must lie on $l^{'}$. The camera baseline intersects each image plane at the epipoles $\mathbf{e}$ and $\mathbf{e}^{'}$. T is the camera transformation between the two views. All epipolar lines intersect at the epipole. Figure adapted from Hartley and Zisserman [2003].

about the scene. The extent of these assumptions largely depends on the method used, what we want to reconstruct, and the required density of the reconstruction. For instance most SLAM or SfM algorithms only model rigid parts of a scene in 3D, and assume that majority of the world is rigid, that the surfaces observed are Lambertian (the apparent brightness of a Lambertian surface to an observer is the same regardless of the observer's angle of view) and/or consist of distinct features that allow them to be matched across images. In the case of *dense* rigid scene reconstruction, additional assumptions are made, for example that the scene structure varies smoothly, or consists of large planar regions.

Another stream of work attempts to model dynamic (moving or deforming) objects in 3D as well. Non-rigid reconstruction given only RGB images from a monocular camera is much more challenging, as in addition to reconstructing the scene, it involves estimating per-pixel trajectories (as opposed to a per-frame trajectory). Without the right amount of data or priors, multiple incorrect solutions can satisfy the geometric model. However much progress has been made towards this direction, e.g., by assuming that the reconstructed objects' shape and trajectory are constrained to some low dimensional manifold (Garg et al. [2013]), while other work simplify the problem by using RGB-D sensory data, and assume that the deformations, if any, are "as-rigid-as-possible" (Newcombe et al. [2015]). Nevertheless, all of these rigid and non-rigid geometric reconstruction methods achieve varying degrees of success depending on the extent of information provided by the input data (e.g. presence of texture in images, availability of a large number of photometrically consistent RGB images, availability of RGB-D sensory data, etc.). More importantly, all of these models are built upon handcrafted assumptions about the scene that may be violated in practice.

The goal in our work is to use deep learning, CNNs in particular, in order to minimise the need for handcrafted assumptions about the scene, the need for handcrafting features for pixel-level dense correspondence between images, and to enable more reliable reconstructions when presented with poor or limited sensory evidence. The expressiveness and automatic end-to-end learning capabilities of CNNs allow them to capture highly non-linear correlations — otherwise difficult to handcraft — that exist between the data and the scene, given a large amount of training data. This learned information could be in the form of pixelwise surface normals and depths, good features to match, confidence of observations, etc. In this thesis we propose how we can make use of such learned information for geometric reconstruction, look at better ways of learning them, and in this process also get to know when and where they are most useful.

Towards this goal, we explore the use of CNNs in the context of dense rigid scene reconstruction. In particular, we first focus our attention to the problem of *keyframe reconstruction* given a set of nearby images and camera poses — this problem is part of the typical front-end in modern SLAM methods that alternate between keyframe reconstruction and camera pose estimation. We provide a more thorough review regarding the typical SLAM front-end and back-end in Chapter 2.

For keyframe reconstruction in the rigid scene case, given a camera transformation T (rotation and translation) between a keyframe (the frame which we want to reconstruct), and a live image, the 3D landmark location of a particular pixel in the keyframe can be found by moving the 3D point along that pixel's viewing ray (i.e. varying the depth/range of the point from the camera centre) and matching the color (or feature) of the point against that of the projected location in the live image.

These projections in the live image all fall in a line called the epipolar line, and the matching problem is conveniently restricted to a line search. Hence there is a one-to-one correspondence between a location along the epipolar line in the live image and (inverse) depth of a pixel in the keyframe. The distance along the viewing ray from the camera centre for a particular pixel in the keyframe, that gives the best color (or feature) match with the projected location in the live image (on the epipolar line) is considered to be correct depth for that keyframe pixel. Figure 1.2 illustrates the aforementioned geometric property that is exploited during the keyframe reconstruction process (a full resource on the various geometric relations pertaining to two-view and multi-view reconstruction is presented in Hartley and Zisserman [2003]).

When provided with multiple live images overlapping a keyframe, for e.g. in the case of a moving camera capturing images at 30 Hz, the matching evidence can be accumulated as a function of (inverse) depth, which is a strategy employed by most dense SLAM systems, particularly the successful dense mapping framework of DTAM (Newcombe et al. [2011]) that forms one of the baselines for our work. DTAM also uses an *inverse depth* smoothness prior in order to regularize the depth of pixels that otherwise have low matching evidence.

One of our contributions lies in replacing this smoothness prior with a surface-normals-based inverse depth regularizer. More importantly these surface normals are learned and directly predicted from the keyframe image via a CNN. Note that we provide a justification as to why inverse-depth parameterization is preferred (over standard depth parameterization) for keyframe reconstruction as we introduce our surface normals-based regularizer in Chapter 3. Figure 1.3 illustrates how our proposed regularization term in Chapter 3 improves over the inverse depth smoothness prior of DTAM.

Another one of our contributions lies in replacing RGB features that are used for matching in DTAM with our large receptive field learned features explicitly trained via a CNN to be "good for matching". Figure 1.4 illustrates how using our learned features to construct the matching cost volume instead of using RGB features enables more accurate matching even in textureless regions, and helps simplify the optimization process by making the data term based on matching cost more convex. Figure 1.5 provides an example of the combined benefits of integrating our learned normal prior with learned good features for matching.

Just-In-Time Reconstruction: Scalability is another major issue in SLAM. Most SLAM systems, for practical purposes resort to managing a sparse map for large scale mapping and localisation; since it is efficient in terms of memory management and computation to work with sparse point clouds. However, dense maps are crucial for reliable navigation and object manipulation. *What if, upon re-visiting a map, we could rapidly create a dense map on-demand based only on a sparse point cloud and some richer representation of a scene derived from a single image?* In fact, some inspiration for this strategy can be drawn from the way humans store and recall a "vague picture" of the geometry of previously visited environments thus making it easier and quicker for us to "adjust" and get a more complete understanding of the environment upon revisiting it.

For some applications, sparse SLAM systems such as ORB-SLAM (Mur-Artal et al. [2015]) suffice, but in other applications — or even within the same applications at different times — there would be significant benefit from having a dense map. It is the possibility to use sparse mapping to efficiently build a "skeleton map" — that is robust and accurate for camera (re-)localisation — on top of which

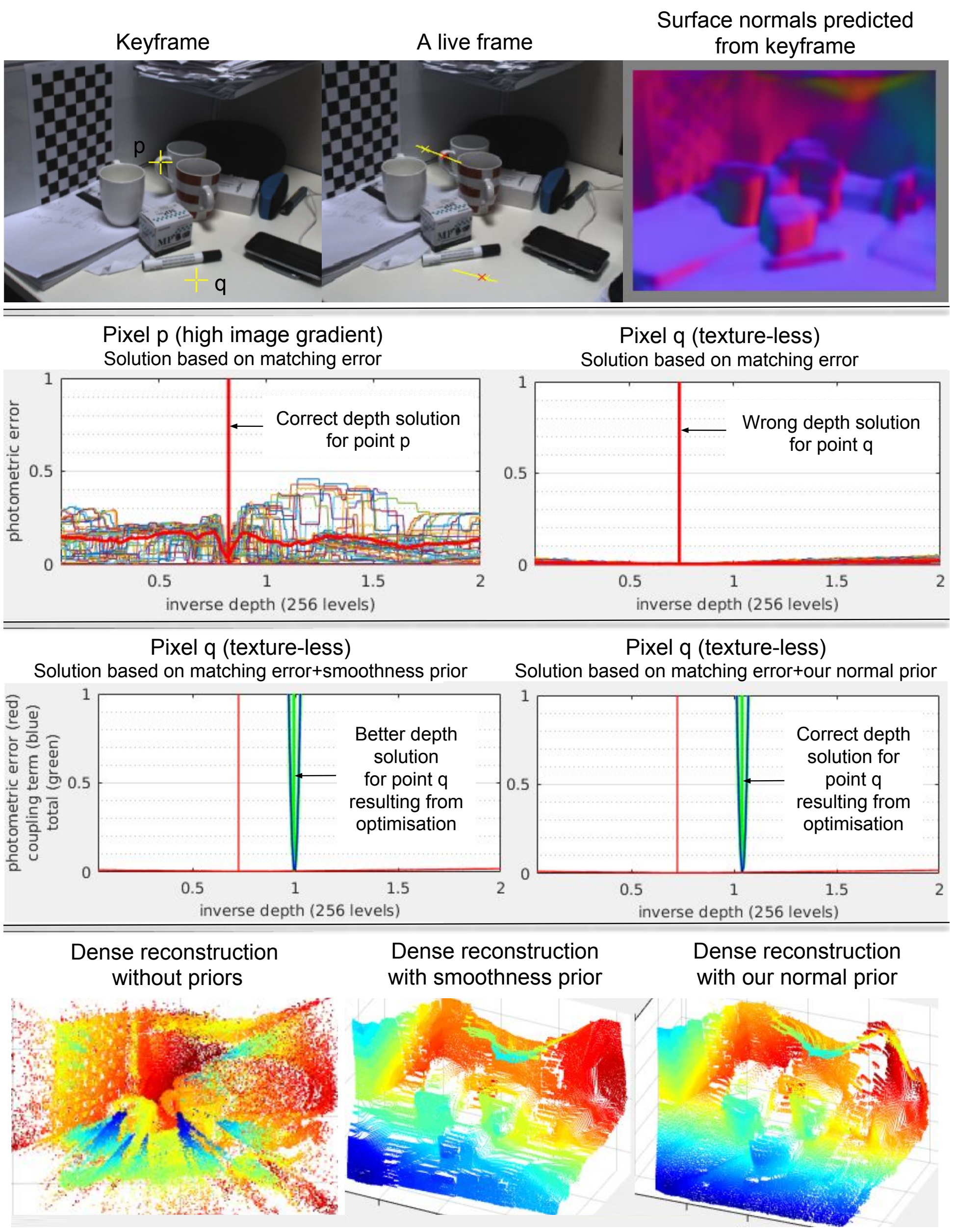


Figure 1.3: The figure illustrates more in depth detail about the matching problem in texture-less regions and how the surface normal prior that we propose in Chapter 3 helps guide the solution towards the ideal minimum, in contrast to the weaker smoothness prior of DTAM (Newcombe et al. [2011]). The yellow lines in the live frame are the corresponding epipolar lines for pixels p and q marked in the keyframe. There is a one-to-one correspondence between location along the epipolar lines and the (inverse) depth solution space of p and q. The yellow cross on each of the epipolar lines indicates the best match based on matching the color of p or q with that of the live frame pixels along the respective epipolar line (a correct match would give the correct depth solution for p or q and vice-versa). The location of the red cross on each of the epipolar lines is inferred from the depth solution obtained via accumulated matching evidence given a series of live images, i.e. the solution inferred from the bold red curve of the plots in the second row. The plots in the third row show the state of the energy and its solution after running the optimisation process with the respective priors. The surface normals are predicted from the keyframe image by our replicated CNN model based on Eigen and Fergus [2015]. The reconstructions are overlayed with pseudo color representing depth.

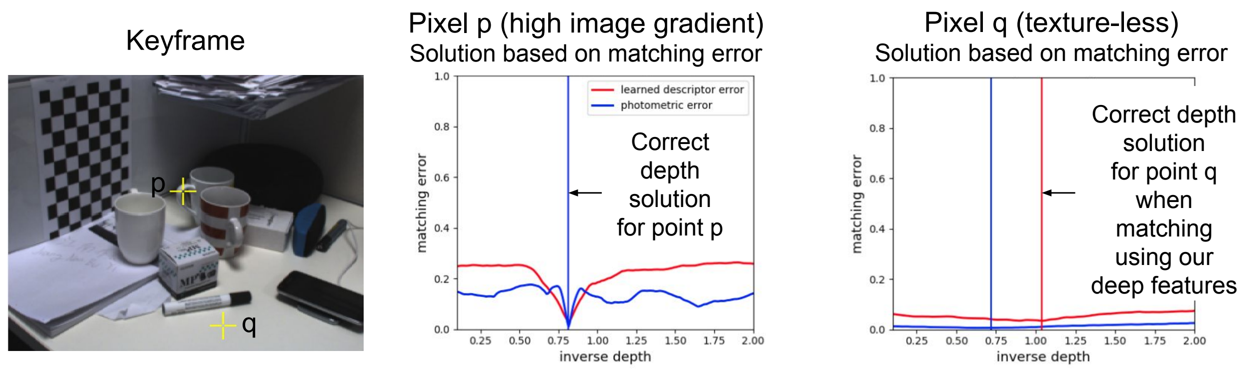


Figure 1.4: The figure illustrates how the learned features that are good for matching which we propose in Chapter 4 enable more reliable matching even for low-image gradient pixels like point q. Notice how the red vertical line points to about the same solution as when using the normal prior in Figure 1.3. Notice also that the graph of matching error vs inverse depth labels is more convex (contains few local minima) when using the learned descriptors for matching, which is more desirable from an optimisation perspective.

we can build a dense map *on demand* — that motivated our "just-in-time" reconstruction strategy (Figure 1.6). Some results of our just-in-time reconstruction approach are shown in Figure 1.7.

In the limit case our "just-in-time reconstruction" framework can "fill in" a sparse map (a bundle adjusted SLAM map / sparse point cloud acquired from a 3D sensor) based on a single image (Chapter 5). In the more general case it could also use a few nearby frames to accumulate photometric evidence, similar to the systems proposed in Chapter 3 and 4, and we discuss more about this unified model at the end of Chapter 5.

1.2.2 Scene Understanding for Geometry Estimation

Depth recovery from a single image is inherently an ill-posed problem, as multiple 3D structural configurations may lead to the same 2D appearance. In practice, however, most naturally occurring or man-made structures exist in a finite set of structural configurations (shapes and sizes), bear common properties, and adhere to a common set of physical constraints. This makes it possible to establish some form of strong correlation between distinctive cues that exist in an image to plausible 3D structure, and generalize based on this — learning such a correlation between a single image and scene structure, is largely believed to be the phenomenon humans and other animals exploit for depth recovery from one eye (Bülthoff et al. [1998]; Loomis [2001]), and the ability to generalize efficiently with a finite model capacity, based on only a sample of data acquired from the world, is in fact the single most challenging task in machine learning, and serves as a performance benchmark for the learned model.

Towards the goal of single-view reconstruction, early methods aimed to manually analyze and extract depth and shape cues from an image, and some of these cues include shading (Zhang et al. [1999]), shadows, contours, texture patterns, vanishing points and lines, depth of field, occlusions, sizes of known objects, etc. What's interesting about single view depth prediction is that it makes no explicit assumption about scene rigidity, making it highly useful as a prior or a "virtual depth sensor" for both rigid and non-rigid reconstruction. One of the early learning based methods to achieve reasonable success was Make3D (Saxena et al. [2006]) which relied upon a set of handcrafted features to map

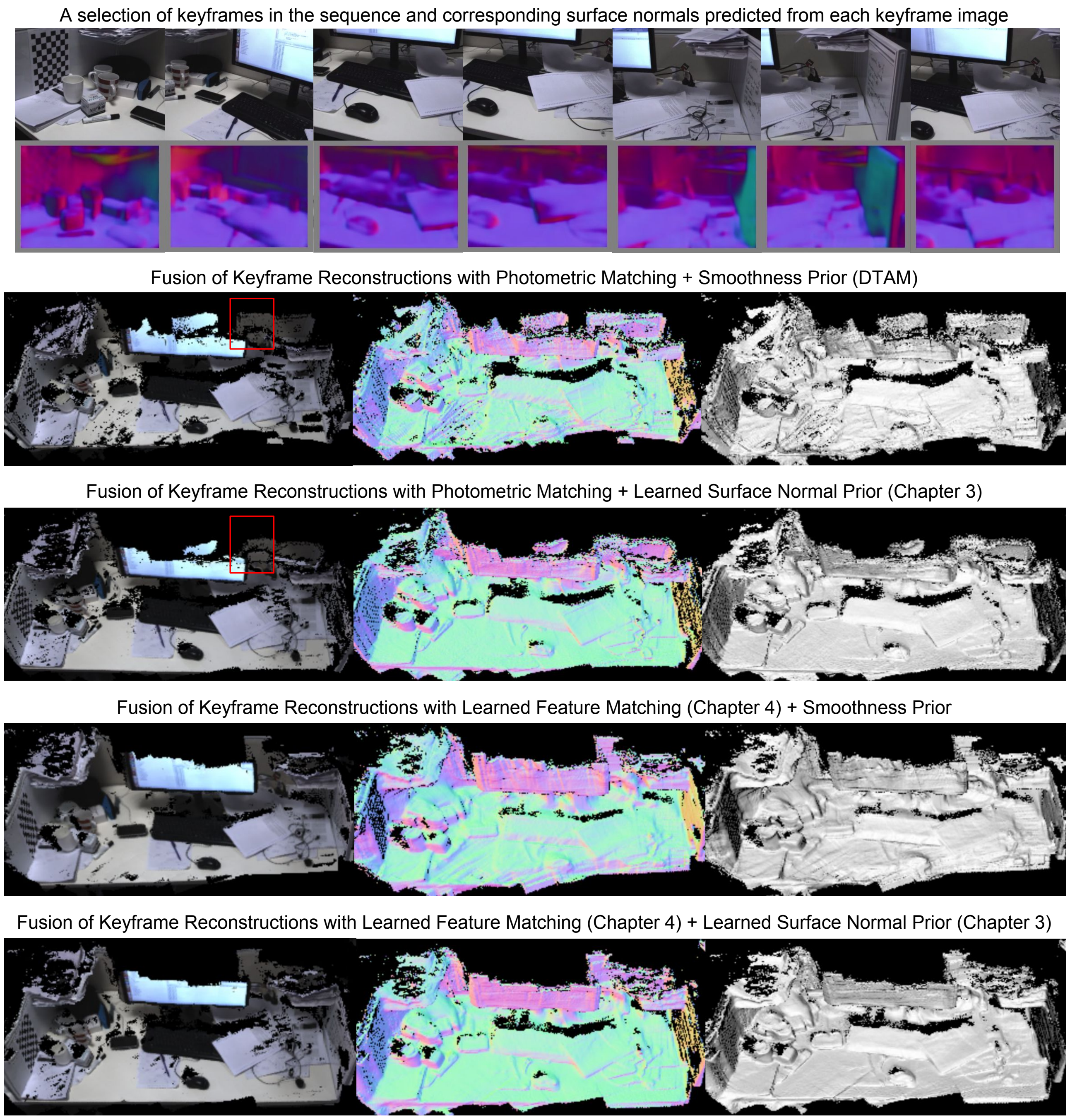

A selection of keyframes in the sequence and corresponding surface normals predicted from each keyframe image
Fusion of Keyframe Reconstructions with Photometric Matching + Smoothness Prior (DTAM)
Fusion of Keyframe Reconstructions with Photometric Matching + Learned Surface Normal Prior (Chapter 3)
Fusion of Keyframe Reconstructions with Learned Feature Matching (Chapter 4) + Smoothness Prior
Fusion of Keyframe Reconstructions with Learned Feature Matching (Chapter 4) + Learned Surface Normal Prior (Chapter 3)

Figure 1.5: The figure illustrates how both our learned surface normal prior (Chapter 3) and learned features that are good for matching (Chapter 4), improve the quality of dense reconstruction. Our surface normal prior greatly improves the reconstruction by respecting the true orientations of the surfaces observed while the smoothness prior typically results in fronto parallel (w.r.t. image plane) surfaces in textureless regions, especially when matching based on photometric error. However both the smoothness prior and our surface normal prior provide limited information about depth relationships at depth discontinuities, hence the textureless wall behind the monitor is reconstructed wrongly (the red rectangle in the first and second reconstruction from top). This issue is resolved using our learned features for matching as seen in the bottom two reconstructions. Normals are predicted using the CNN model based on Eigen and Fergus [2015]. The large black holes in the overhead view of all the reconstructions correspond to regions unobserved by the camera.

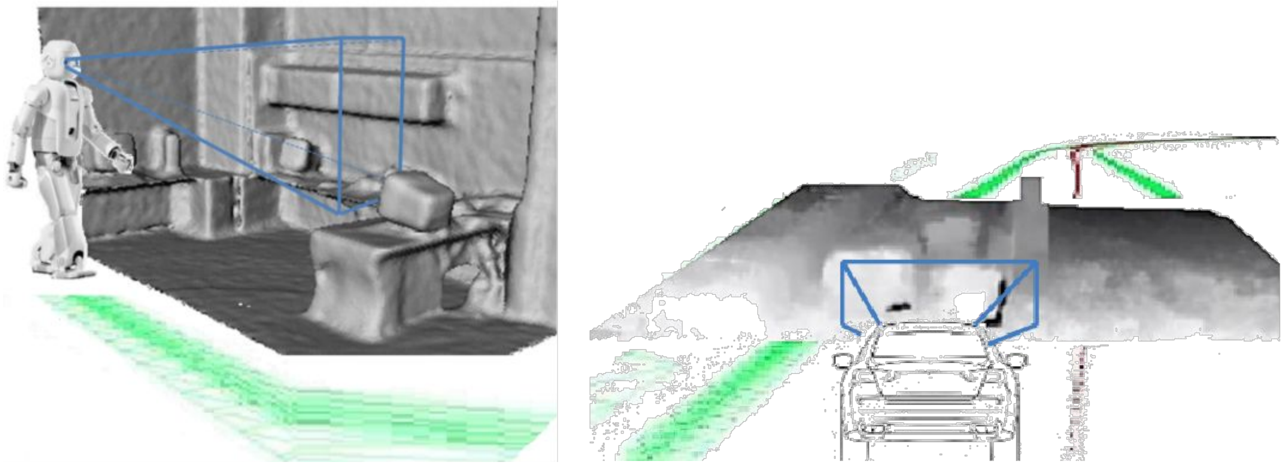


Figure 1.6: Conceptualization of "Just-in-Time Reconstruction". The green lines represent some compact representation of the map, while the Phong-shaded regions show the "just-in-time" dense reconstructions computed based on the compact map and a combination of high-level and low-level geometric scene understanding.

patches from the input image to some feature space, and in turn learn to regress from that feature space to depth values. They also use an additional pairwise depth smoothness prior, modeled as either a Gaussian or Laplacian distribution with a data dependent learned variance, in order to regularize and globally optimize for a depth map, with the belief that the process of recovering a depth map requires global reasoning on the image.

Great progress has been since then, and much of this progress is attributed to the powerful feature learning capabilities of Convolutional Neural Networks (CNNs) that, given enough data, model capacity, and model training, can not only automatically learn to extract suitable features from an image, but also learn highly non-linear mappings from these extracted features to the desired output, either with (Liu et al. [2015b]; Wang et al. [2016]) or without (Eigen and Fergus [2015]) an explicitly defined scene prior model. The multi-scale architecture proposed by Eigen and Fergus [2015] in particular, for depth, surface normal, (and pixelwise semantic label) estimation from a single image is much efficient than previous CNN-based methods that involved more engineering and/or required superpixel extraction and patch-based regression (Liu et al. [2015a]; Wang et al. [2014]).

Figure 1.8 shows Eigen and Fergus [2015]'s neural network architecture which we exploit for extraction of both surface normals and depth from a single image, for the experiments carried out in this thesis involving indoor data. Our Caffe (Jia et al. [2014]) implementation of their network can predict a depth map and surface normal image in real-time on a commodity GPU ($\approx 40ms$ on a Nvidia GTX 980). Note however that any modern CNN architecture for surface normal and depth prediction can be integrated into our work. It is also interesting to note the recent work in unsupervised learning of depths from a single image without need for availability of groundtruth sensory data for training (Garg et al. [2016]), which is especially useful given limited or sparse training depth data on outdoor datasets. We use the model proposed by Garg et al. [2016] for depth prediction on outdoor data. In this thesis, we propose scene prior models that incorporate learned geometry information via CNNs in the form of surface normals (Chapter 3) and depths (Chapter 5), in order to make use of this information for geometric reconstruction.

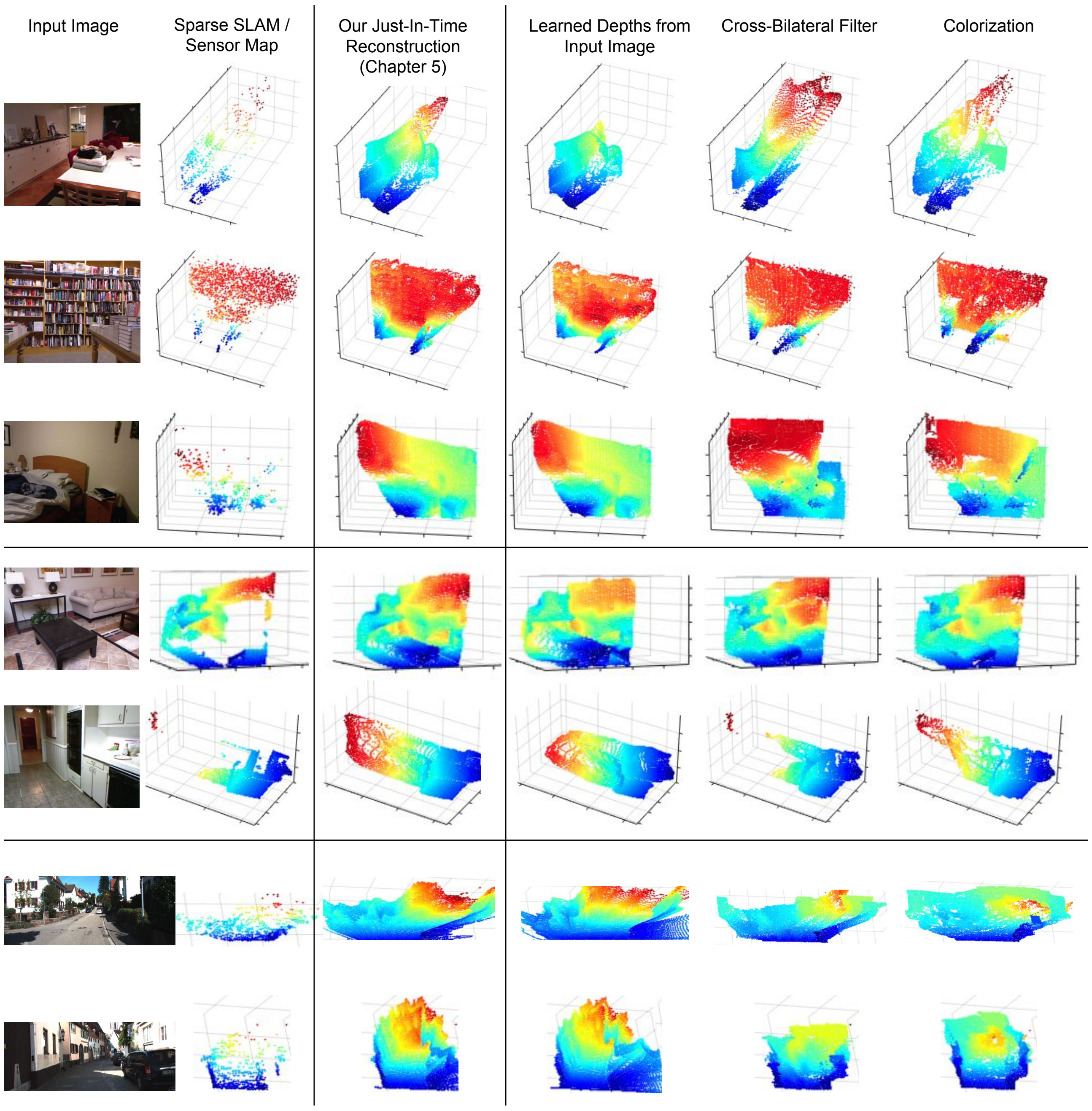

Input Image
Sparse SLAM / Sensor Map
Our Just-In-Time Reconstruction (Chapter 5)
Learned Depths from Input Image
Cross-Bilateral Filter
Colorization

Figure 1.7: The figure illustrates results of our proposed "just-in-time" reconstruction framework in Chapter 5, along with results of our three baselines, i.e., learned single-view depths from Eigen and Fergus [2015]; Garg et al. [2016] for indoor/outdoor scenes, the cross-bilateral filter (Tomasi and Manduchi [1998]), and Colorization (Levin et al. [2004]). The latter two methods rely on low-level local color similarity information from the input image to propagate depths from the sparse map, while our method uses learned depths (and confidences) as described in Chapter 5, for more reliable inpainting, especially when the amount of sparsity is high. Note that the monocular SLAM maps produced by ORB-SLAM (Mur-Artal and Tardós [2016]) in the first three rows (1) contain outliers and (2) are not in groundtruth (metric) scale and hence not in the same scale of the single view depth predictions (although the depth maps shown here are manually scaled to groundtruth scale for easier visualization). Both these issues are accounted for in our confidence-based scale-invariant depth fusion framework in Chapter 5. The sparse maps in the middle section rows are acquired by the Kinect sensor, while those from the last two rows are acquired by the LIDAR sensor. The Kinect and LIDAR depth data (which is already either semi-dense or sparse) was further sparsified manually (while respecting their unique sparsity pattern), to aid evaluation.

Relation to Conditional Random Fields (CRFs): It is quite common in computer vision to model a problem as inference of an energy function consisting of data terms and regularization terms, corresponding to the observation model and prior model respectively:

$$E(\mathbf{y}) = E_{data}(\mathbf{y}) + E_{reg}(\mathbf{y}) \tag{1.1}$$

This energy can be associated to or derived from the negative log likelihood of the following posterior (or conditional) probability distribution, $E(\mathbf{y}) = -log(P(\mathbf{y}|\mathbf{x}))$, where:

$$P(\mathbf{y}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{y})P(\mathbf{y})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\mathbf{y})P(\mathbf{y})}{\int_{\mathbf{y}} P(\mathbf{x}|\mathbf{y})P(\mathbf{y})d\mathbf{y}} = \frac{exp(-E(\mathbf{y}))}{\int_{\mathbf{y}} exp(-E(\mathbf{y}))d\mathbf{y}} \tag{1.2}$$

Notice that the partition function $Z = \int_{\mathbf{y}} exp(-E(\mathbf{y}))d\mathbf{y}$ is usually ignored during inference as it is independent of (marginalized over) $\mathbf{y}$. However this normalization function becomes important when learning the model parameters such as variances (as we see in Chapters 3, 4, and 5).

$P(\mathbf{x}|\mathbf{y})$ models the distribution of observations given $\mathbf{y}$ as some function of $\mathbf{y}$ (observation model), and $P(\mathbf{y})$ models the distribution of $\mathbf{y}$ (prior model). $P(\mathbf{y})$ is generally approximated by some assumption (prior) regarding interdependencies of output variables, and in the case of DTAM, $E_{reg}(\mathbf{y}) = -ln(P(\mathbf{y})) = |\nabla\mathbf{y}|$, i.e. a prior formulated based on the assumption that the inverse depth values are spatially smooth. The optimal solution $\mathbf{y}^*$ can be found by minimizing $E(\mathbf{y})$, which is equivalent to obtaining the maximimum a posterior estimate for $\mathbf{y}$ that maximizes $P(\mathbf{y}|\mathbf{x})$.

A conditional (or discriminative) model such as this, whose conditional probability distribution $P(\mathbf{y}|\mathbf{x})$ is typically modeled based on the dependency of output variables on the observations and the inter-dependencies of the output variables themselves (in the form of vertices and edges of a graph) is known as a Conditional Random Field (CRF).

Our proposed models for keyframe reconstruction in this thesis are based around maximizing this conditional probability, and in our case the variable $\mathbf{y}$ represents the (inverse/log) depth values of pixels in the keyframe image, and $\mathbf{x}$ represents the observations which are for example a set of input images (and corresponding camera poses), a noisy and sparse map, and/or sensory data from a depth sensor. *We focus on improving both the observation and prior models.*

For the majority of this thesis (Chapter 3 and Chapter 5), we focus on better ways of modeling E_{reg}, i.e. $P(\mathbf{y})$, primarily by exploiting learned normals and depths as priors, thus eliminating the need for the usual structural smoothness assumption about the scene. In Chapter 4 we focus on improving the reliability of the data term E_{data}, i.e. better model for $P(\mathbf{x}|\mathbf{y})$, by learning CNN features that are "good for matching".

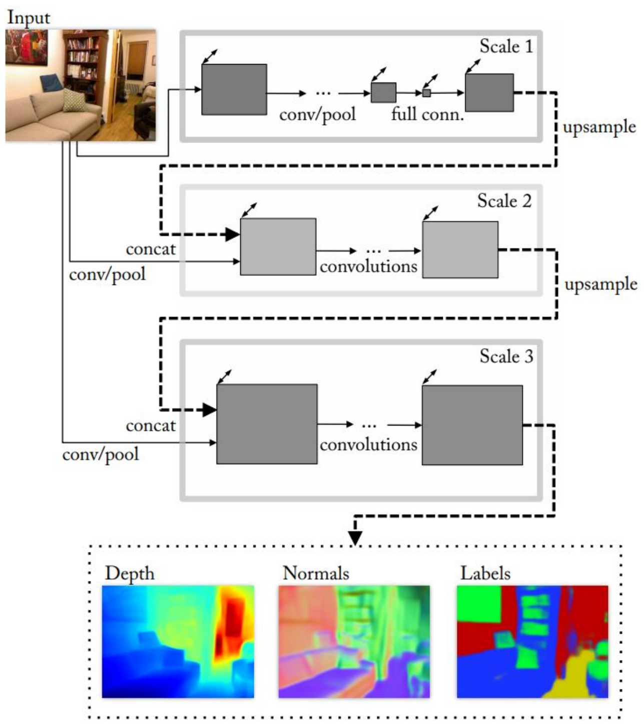


Figure 1.8: Multi-scale CNN architecture from Eigen and Fergus [2015] for regressing a depth map, normal map, and semantic label map from a single image.

1.3 Contributions

This thesis presents the following main contributions:

C1 We propose a novel anisotropic surface regularizer where the magnitude of penalization is the confidence-weighted *norm* of a simple continuous and linear function involving the surface normal and pairwise inverse depth gradient at each pixel of a keyframe, and is formulated based on projective geometry. Being a convex energy function, it can be minimised quite easily with standard optimisation methods, and in our experiments we use the efficient first order primal-dual formulation proposed by Chambolle and Pock [2010], and subsequently used in DTAM (Newcombe et al. [2011]). We show that our anisotropic formulation can favor arbitrary surface orientations as indicated by the per-pixel surface normals, and that it is a generalization of the inverse depth smoothness prior of DTAM which favour fronto-parallel planes. When minimised in combination with a data term like sum of photometric error in the same framework of DTAM, we obtain an inverse depth solution that (1) is consistent with the pixelwise normals, and (2) satisfies multi-view photometric consistency. Therefore this combined formulation can be seen as an efficient energy minimization-based method for surface normal to depth integration that also is multi-view photoconsistency guided. (Chapter 3, Section 3.3.2).

C2 Based on the above formulation, we present the first study of the effects of using learned surface normals predicted from a Convolutional Neural Network (CNN) in the context of real-time live dense monocular reconstruction. (Chapter 3, Section 3.4).

C3 Along with several other future directions for the above work, we propose a potential scheme to learn the pixelwise confidences of our surface normal-based pairwise energy terms. We do so by modeling the pairwise terms as a multi-variate Gaussian distribution and learning the pixelwise variances (inverse confidences) as a function of the keyframe image such that the likelihood is maximized. These learned confidences may then replace the image-edge-based confidence weights — a heuristic used typically to represent confidences of the regularization terms and minimize penalization at depth discontinuities. (Chapter 3, Section 3.5.2).

C4 We propose a novel self-supervised training methodology and CNN model for learning large-receptive field deep features that are good for matching. When compared with the matching performance of traditional dense features like affine-warped image patches and dense SIFT, as well as Conv1 features from an ImageNet pretrained ResNet-50 network (He et al. [2016]), our learned features generalize better under a variety of appearance changes, while being very fast to extract. (Chapter 4, Section 4.3).

C5 We provide the first study of the use of CNNs in the context of live dense monocular reconstruction for *both* dense correspondence (in the form of good features to match) and geometric scene understanding (in the form of learned normals). (Chapter 4, Section 4.4.1).

C6 We show that when our learned features are used as a feature reconstruction loss in unsupervised depth estimation methods like Garg et al. [2016], on top of the photometric error-based loss, we obtain improved depth predictions. In the case where groundtruth depth is unavailable we show that depth predictions can be used as proxy groundtruth for learning good features to match, and these features in turn can improve depth predictions when used in the feature reconstruction loss for unsupervised depth estimation. (Chapter 4, Section 4.4.2).

C7 As a future direction, we propose a method to learn the confidences of the multi-view observation model parameterized by our learned features for matching, on a per-pixel basis. This is helpful as it will enable us to know which pixels our learned features are best at matching, and allow us to discard/minimize the impact of potentially wrong matches for dense mapping and tracking. To do so, we approximate the matching probability distribution for each pixel in the keyframe along the epipolar line with a Gaussian distribution with a mean equal to the inverse depth error corresponding to the global minimum matching error, and then *learn* its variance as a function of the keyframe image such that the likelihood is maximized. (Chapter 4, Section 4.5.1).

C8 Additionally, the reliable and unique matching properties of our learned features enable us to approximate the non-convex matching error energy function for each pixel in the keyframe (which is parameterized by inverse depth) with a convex energy function based on the learned Gaussian distribution from C7. The convex data term with our convex prior enables theoretical convergence guarantees for rapid dense multi-view keyframe reconstruction. (Chapter 4, Section 4.5.2).

C9 We propose a novel confidence-weighted scale-invariant pairwise depth energy function which is convex with respect to the log-depth map solution of the keyframe, and when minimized involves satisfying *learned* pairwise log-depth relationships, based on their *learned* pairwise confidences. We use this learned depth prior to inpaint partial depth maps for an image (for e.g. obtained from mono SLAM, or a sparse 3D sensor), outperforming traditional depth inpainting methods such as the cross-bilateral filter (Petschnigg et al. [2004]) and colorization (Levin et al. [2004]) that are based on handcrafted priors. We model the problem as inference over a fully connected CRF, with edges extracting depth relationships from learned depth predictions, and nodes anchoring the solution to the sparse points. Both the nodes and edges of the CRF are also parameterized by pixelwise *learned relative depth confidences* (confidences relative to other depth values in the same depth map) which are used to reduce penalization by energy terms involving uncertain depths. A simple relaxation of the pairwise confidence weights (i.e. $c^d_{ij} = c^d_i c^d_j$) reduces the number of parameters from $O(N^2)$ to $O(N)$, N being the number of pixels, and allows for efficient linear time inference. Importantly, we learn to predict the confidence weights of our CRF model directly from the input image and partial depth map, based on a training signal that measures the scale invariant depth error for each input depth map with respect to the groundtruth depth map. Our formulation which is quadratic can be solved efficiently with existing convex optimization methods, and in our experiments we use the iterative conjugate gradient decent algorithm and parallelize parts of it on the GPU. (Chapter 5, Section 5.2).

C10 Using the above formulation, we advocate the concept of "just-in-time" reconstruction where a compact representation of a 3D map (in our case in the form of a sparse map) is densified on-demand using a higher level understanding of the scene upon revisiting the map (in our case in the form of a learned depth map and learned confidence maps based on a single view). The benefit of this approach is that we do not have to store and maintain a dense map which is too expensive for practical reasons, while on the other hand we do not need to reconstruct the map from scratch upon revisiting it (since often the environment remains the same for the most part). This allows for compact storage, and fast and reliable reconstruction upon revisiting a map even with a single view. (Chapter 5).

C11 Among several future directions for the above work we propose a more ideal means for learning pixelwise *relative* confidences of depth values in a depth map (of arbitrary depth scale and sparsity structure) by modeling the pairwise scale-invariant depth error energy terms as a multivariate Gaussian distribution (simplified with a pairwise confidence weight relaxation resulting in only $O(N)$ learnable parameters as before), and learning the pixelwise variances as a function of the keyframe image that maximize the likelihood. In the more general case, this confidence learning scheme can be applied to any continuous multivariate measurement signal (without explicit data pre-processing) where the measured variables and associated groundtruth are not necessarily in the same scale, the measured variables consist of outliers of unknown degree, and knowing the relative confidences between the measured variables (as opposed to knowing absolute confidences) is sufficient. (Chapter 5, Section 5.4.3).

C12 Finally, we present our probabilistic scale-invariant pairwise depth energy function in *inverse depth* parameterization (respect to the keyframe) so that it can be (more easily) *jointly* minimized alongside our surface normal-based pairwise energy function (also parameterized in inverse depth). (Chapter 5, Section 5.4.2). Furthermore, inspired by the cross-bilateral filter and colorization methods, we propose an additional depth prior based on learned pairwise pixel affinities. (Chapter

5, Section 5.4.1). To this end we propose a more accurate and unified model for "just-in-time" keyframe reconstruction that *jointly* incorporates prior information (regressed from a keyframe) involving learned depths, learned surface normals, learned pairwise pixel affinities, and multi-view observation information based on learned good features to match (if given one or more overlapping live frames), in addition to the stored sparse depth map observation information. All the terms in this unified model are parameterized by their *learned confidences* and the overall energy is a *convex function* with respect to the *inverse depth solution* of a keyframe. (Chapter 5, Section 5.4.4).

The developed methods will be useful in diverse automated tasks, for e.g., detailed 3D mapping of objects and environments, autonomous navigation and interaction, augmented reality, etc. Since efficiency and parallelization of the algorithms was a key focus, in future, the algorithms may be ported for use in low-power GPU-enabled mobile platforms.

1.4 Thesis Outline

The following is an outline for the rest of this thesis:

- **Chapter 2:** In this chapter we discuss sparse and dense SLAM methods, image correspondence estimation methods, Convolutional Neural Networks and their use for estimating geometry from an image, and Conditional Random Fields and their use for modeling priors for dense reconstruction. We conclude with a mathematical background on the convex conjugate and the primal-dual optimisation algorithm in relation to our formulation in Chapters 3 and 4.

- **Chapter 3:** In this chapter we introduce our learned surface normals-based prior for dense keyframe reconstruction, alongside a framework for live dense monocular reconstruction. This chapter encompasses contributions: C1, C2, C3.

- **Chapter 4:** This chapter introduces our CNN model and training methodology for learning good features for dense correspondence between images. We integrate these features into our framework in Chapter 3 where they replace raw photometric data for multi-view matching. We analyze the combined benefits of having a learned normal prior and learned features that are good for dense matching in the context of dense multi-view reconstruction. This chapter encompasses contributions: C4, C5, C6, C7, C8.

- **Chapter 5:** In this chapter we introduce our framework for "just-in-time reconstruction", that uses single view depth predictions as priors for reconstruction. We provide an extensive study on the benefits of our framework in comparison with traditional depth in-painting methods. We conclude this chapter with several possible extensions to our formulation, including a unified model for geometric reconstruction that combines the ideas/models proposed in this chapter with those of Chapters 3 and 4. This chapter encompasses contributions: C9, C10, C11, C12.

- **Chapter 6:** This chapter provides an overall summary of the work in this thesis and a discussion of some avenues for future work, following up on those discussed in the previous chapters.

Bibliography

I. Bülthoff, H. Bülthoff, and P. Sinha. Top-down influences on stereoscopic depth-perception. *Nature neuroscience*, 1(3):254, 1998.

A. Chambolle and T. Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2010.

D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

R. Garg, A. Roussos, and L. Agapito. A variational approach to video registration with subspace constraints. *International journal of computer vision*, 104(3):286–314, 2013.

R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. ISBN 0521540518.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23:689–694, 2004.

F. Liu, C. Shen, and G. Lin. Deep Convolutional Neural Fields for Depth Estimation from a Single Image. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015a. URL http://arxiv.org/abs/1411.6387.

F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *Technical report, University of Adelaide*, 2015b. URL http://arxiv.org/abs/1502.07411.

H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133, 1981.

J. M. Loomis. Looking down is looking up. *Nature*, 414(6860):155, 2001.

R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, abs/1610.06475, 2016. URL http://arxiv.org/abs/1610.06475.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

D. Nister. An efficient solution to the five-point relative pose problem. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–195. IEEE, 2003.

G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. In *ACM transactions on graphics (TOG)*, volume 23, pages 664–672. ACM, 2004.

A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille. SURGE: Surface Regularized Geometry Estimation from a Single Image. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 172–180. 2016.

X. Wang, D. F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. *CoRR*, abs/1411.4958, 2014. URL http://arxiv.org/abs/1411.4958.

R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analaysis and Machine Intelligence*, 21(8):690–706, 1999.

Chapter 2

Literature Review

This chapter aims to provide a more detailed review on multi-view geometric reconstruction and scene understanding for geometric reconstruction in line with the main focus areas of this thesis. We discuss sparse and dense SLAM methods, image correspondence estimation methods, Convolutional Neural Networks and their use for estimating geometry from an image, and Conditional Random Fields and their use for modeling priors for dense reconstruction. In the remaining part of this chapter, we provide a mathematical background on the convex conjugate and the primal dual optimization method in relation to our formulation in Chapters 3 and 4.

2.1 Multi-View Geometric Reconstruction

2.1.1 Offline Structure from Motion (SfM)

The concept of using multi-view images to recover the 3D structure of a scene, while also recovering the camera poses at which the images were captured, involves solving an old and well-defined problem in computer vision called Structure from Motion (SfM). This concept also appeared early on in the photogrammetry and geodesy literatures (Kraus [2007]). Bundle adjustment is the gold standard for approaching the SfM problem (Triggs et al. [1999]). The name refers to the 'bundles' of light rays leaving each 3D feature and converging on each camera centre, which are 'adjusted' optimally with respect to both feature and camera positions. Equivalently — unlike independent model methods, which merge partial reconstructions without updating their internal structure — all of the structure and camera parameters are adjusted together 'in one bundle' (Triggs et al. [1999]).

Probabilistically, the structure from motion problem can be viewed as finding the camera poses and 3D landmark coordinates that maximize the following conditional probability:

$$P(\mathbf{T}_{0:k}, \mathbf{x} | \mathbf{Z}_{0:k}, \mathbf{T}_0) \tag{2.1}$$

where, $\mathbf{T}_{0:k} = \{\mathbf{T}_0, ..., \mathbf{T}_k\} = \{\mathbf{T}_{0:k-1}, \mathbf{T}_k\}$ denote the set of unknown camera locations and orientations, $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ denote the set of all landmark locations, and $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$ denote the set of landmark observations. $\mathbf{T}_0$ is set as the reference (world) coordinate frame in order to help mitigate the gauge freedom (Triggs et al. [1999]).

Estimating the optimal $\mathbf{T}_{0:k}$ and $\mathbf{x}$ via bundle adjustment is equivalent to maximizing the likelihood (minimizing the negative log likelihood) of the above conditional probability. This usually involves solving a non-linear least squares estimation problem using the Levenberg-Marquardt algorithm (Triggs et al. [1999]) which *jointly* optimizes for the poses and landmark locations that minimize the sum of re-projection errors (a process also referred to as *smoothing* in the SLAM community):

$$E_{data}(\mathbf{T}_{0:k}, \mathbf{x}) = \sum_{i,j} \rho_h(\mathbf{e}_{i,j}^T \Omega_{i,j}^{-1} \mathbf{e}_{i,j}) \tag{2.2}$$

where ρ_h is a robust cost function, e.g. Huber function. The indices i and j correspond to camera/(key)frame index and landmark index respectively. $\Omega_{i,j}$ is the covariance matrix representing uncertainty of the observation of landmark j in camera i. The re-projection error $\mathbf{e}_{i,j}$ is usually measured in 2D pixel space as a distance from the 2D location of the true match $\mathbf{u}_{i,j}$ for landmark j in camera i:

$$\mathbf{e}_{i,j} = \mathbf{u}_{i,j} - \pi_i(\mathbf{T}_{iw}, \mathbf{x}_{w,j}) \tag{2.3}$$

where π_i is the projection function:

$$\pi_i(\mathbf{T}_{iw}, \mathbf{x}_{w,j}) = \begin{bmatrix} f_{i,u}\frac{x_{i,j}}{z_{i,j}} + c_{i,u} \\ f_{i,v}\frac{y_{i,j}}{z_{i,j}} + c_{i,v} \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} x_{i,j} & y_{i,j} & z_{i,j} \end{bmatrix} = R_{iw}\mathbf{x}_{w,j} + \mathbf{t}_{iw}. \tag{2.5}$$

$R_{iw} \in \mathrm{SO}(3)$ and $\mathbf{t}_{iw} \in \mathcal{R}^3$ are, respectively, the rotation and translation parts of $\mathbf{T}_{iw} \in \mathrm{SE}(3)$ which describe the transformation of a point from the world coordinate frame into the coordinates of camera i, and $(f_{i,u}, f_{i,v})$ and $(c_{i,u}, c_{i,v})$ are the focal length and principle point associated with camera i. $\mathbf{x}_{wj}$ is the 3D location of landmark j with respect to the world coordinate frame. In this instance $\Omega_{i,j}$ can be defined as $\Omega_{i,j} = \sigma_{i,j}^2 I_{2\times 2}$.

Note that the re-projection error $\mathbf{e}_{i,j}$ can also be measured in terms of photometric (or feature) error (Alismail et al. [2016b]). This increases the non-convexity of the problem, but avoids the explicit error inducing data-association step. Using more robust features such those learned using our method in Chapter 4 (weighted by our learned confidences), would make the latter feature-based "direct" bundle adjustment approach converge to a better solution more rapidly.

2.1.2 Visual Simultaneous Localization and Mapping (SLAM)

Being an offline batch optimization method, structure from motion had limited application in real-time tasks. However, sequential and real-time methods for structure from motion were later developed (Davison et al. [2007]; Durrant-Whyte and Bailey [2006]; Harris and Pike [1988]) giving rise to the synonymous term visual SLAM (Simultaneous Localisation and Mapping), a problem of particular interest in robotics. The visual SLAM problem arises whenever a robot is placed in an unknown environment and is expected to move and/or interact inside it, making use of visual information *as and when* captured with *optimal efficiency*. Previously, sensors such as laser range finders were predominantly used for SLAM, but using a camera as the only input (i.e. visual SLAM) has its advantages, e.g. compact size, power efficiency, ubiquity, etc., and the fact that rich geometric information can be extracted even from a single image. Recently bio-inspired event cameras have gotten traction in the vision community due to their fast response time in the order of microseconds in capturing events (changes in intensity) in an asynchronous manner, allowing for High Dynamic Range intensity capture and resilience to motion blur whilst being efficient in terms of data transfer. While conceptually similar, in this review we mainly focus on conventional camera-based methods, hence we point to Kim [2017] for an introduction to event camera-based SLAM methods.

Over the recent decades, much progress has been made in the area of visual SLAM. According to Durrant-Whyte and Bailey [2006] (at that time), at a theoretical and conceptual level SLAM can be considered a solved problem, with the few remaining issues including computation, feature representation, and data association. While exact requirements are largely application dependent, in an ideal scenario, it is desirable for a SLAM system to be capable of all of the following: (1) accurate and dense 3D mapping and precise camera localisation, (2) large-scale operation, (3) real-time low latency execution, (4) long-term consistent performance, (5) static scene mapping in dynamic environments (while modelling dynamic objects if necessary), (6) efficient 3D model representation, and (7) effective utilization of learned priors and stored compact prior scene knowledge for rapid and high-quality reconstruction. Current monocular visual SLAM systems in particular don't satisfy *all* of these desired attributes. These desired attributes are infact inter-related, and in this thesis we mainly focus on attribute (7) while aiming to complement/not compromise the rest of the attributes.

While the first monocular visual SLAM systems were predominantly filter-based, modern SLAM systems are mainly keyframe-based. The advantage of keyframe-based methods over filter-based ones lie mainly in terms of computational cost (Strasdat et al. [2010a]). Another advantage of keyframe-based approaches is that they are less susceptible to linearization errors that accumulate upon state propagation due to the assumption of Gaussian noise in the observation and motion models (as discussed further below). Filter-based methods maintain the state of the landmarks and the *current* pose (the state of previous poses is marginalized out) and explicitly compute and retain the joint state covariance matrix. The computational cost is cubic in terms of the size of the state vector.

Keyframe-based methods retain the state of the whole camera trajectory corresponding to *keyframes* (for the back-end optimization). As highlighted in Strasdat et al. [2010a] retaining the whole trajectory state (i.e. not marginalizing over previous poses) results in overall fewer interdependencies between landmarks and poses that *need* to be modeled for the purpose of bundle adjustment (Figure 2.1) — note that in this thesis we still model the inter-depencies of pairwise landmarks for the purpose of enforcing priors on the dense reconstruction. The sparsity in the state covariance matrix of keyframe-based methods (and not needing to explicitly update the covariance matrix) mean that fewer computations are needed to reconstruct the same number of landmarks. The available computation can be used for dense bundle adjustment, at a trade off of not using information from all frames which do not contribute to map accuracy (Strasdat et al. [2010a]). The front-end of modern SLAM systems however use information from all frames for the purpose of camera tracking and also mapping, for e.g. in the case of DTAM (Newcombe et al. [2011b]) and in our frameworks where the cost volume uses around 30 to 100 of nearby overlapping frames. The recovered poses and landmark coordinates are passed on as initialization to the back-end for further refinement. The success of this front-end and back-end arrangement of keyframe-based approaches has led to its widespread adoption in modern SLAM systems.

Filter-Based SLAM: Given a series of sensor observations over discrete time steps and the control vector used to drive the robot, the goal is to compute an estimate of the robot's location and orientation and a map of the environment. There is almost always uncertainty associated with all quantities, so the objective is to compute and maximize the following conditional probability at a given time step k:

$$P(\mathbf{T}_k, \mathbf{x}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0) \tag{2.6}$$

where $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k\}$ denote the set of control vectors, which may or may not be present. We can maximize the above probability via a standard two-step recursive (sequential) prediction (time-update) correction (measurement-update) form. The time-update step is as follows:

$$P(\mathbf{T}_k, \mathbf{x}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{T}_0) = \int P(\mathbf{T}_k|\mathbf{T}_{k-1}, \mathbf{u}_k) \times P(\mathbf{T}_{k-1}, \mathbf{x}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{T}_0) d\mathbf{T}_{k-1} \tag{2.7}$$

and the measurement update is as follows:

$$P(\mathbf{T}_k, \mathbf{x}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0) = \frac{P(\mathbf{z}_k|\mathbf{T}_k, \mathbf{x}) P(\mathbf{T}_k, \mathbf{x}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{T}_0)}{P(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \tag{2.8}$$

In Equation (2.7), $P(\mathbf{T}_k|\mathbf{T}_{k-1}, \mathbf{u}_k)$ is called the motion model and is described in terms of a probability distribution on state transitions. The state transition is assumed to be a Markov process in which the next state $\mathbf{T}_k$ depends only on the immediately preceding state $\mathbf{T}_{k-1}$ and the applied control $\mathbf{u}_k$ is independent of both the observations and the map. Note that in a purely vision-based SLAM system, the control vector $\mathbf{U}_{0:k}$ may be absent.

In Equation (2.8), $P(\mathbf{z}_k|\mathbf{T}_k, \mathbf{x})$ is called the observation model and it describes the probability of making an observation $\mathbf{z}_k$ when the robot location and landmark locations are known. It is reasonable to assume that once the camera location and map are defined, observations are conditionally independent given the map and the current camera pose state.

Solutions to the probabilistic SLAM problem involve finding an appropriate representation for both the observation model and motion model that allows efficient and consistent computation of the prior and posterior distributions in Equations (2.7) and (2.8) respectively. By far, the most common representation is in the form of a state-space model with additive Gaussian noise, leading to the use of the extended Kalman filter (EKF) to solve the SLAM problem.

Due to the dependency of the observations on both $\mathbf{T}_k$ and $\mathbf{x}$, it follows that the joint posterior cannot be partitioned in the obvious manner:

$$P(\mathbf{T}_k, \mathbf{x}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0) \neq P(\mathbf{T}_k|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0) P(\mathbf{x}|\mathbf{Z}_{0:k}) \tag{2.9}$$

However the SLAM problem has more structure than immediately obvious from the update equations. When reconstructing a rigid scene, even if the pose is estimated incorrectly, the *relative locations* of landmarks $(\mathbf{x}_j - \mathbf{x}_i)$ can be estimated with high certainty (even though its absolute locations may have high variance). The variance of the relative locations of landmarks decreases monotonically as more and more observations are made (at least for the case where we assume the errors in the observation and motion models are Gaussian distributed), leading to more reliable pose estimates. This means that both the pose and landmark location estimates will continue to improve over time (Durrant-Whyte and Bailey [2006]).

One important alternative representation is to describe the camera motion model as a set of samples of a more general non-Gaussian probability distribution. This leads to the use of the Rao-Blackwellized particle filter, or FastSLAM algorithm, to solve the SLAM problem.

The FastSLAM algorithm, introduced by Montemerlo et al. [2002], marked a fundamental conceptual shift in the design of recursive probabilistic SLAM. Previous efforts focused on improving the performance of EKF-SLAM, while retaining its essential linear Gaussian assumptions. FastSLAM, with its basis in recursive Monte Carlo sampling, or particle filtering, was the first to directly represent the nonlinear process model and non-Gaussian pose distribution.

The joint posterior state in FastSLAM may be factored into a camera trajectory component and a conditionally independent map component as in Equation (2.10). Multiple samples of the entire camera trajectory is used to approximate $P(\mathbf{T}_{0:k}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0)$. FastSLAM still linearizes the observation model assuming Gaussian noise in order to estimate $P(\mathbf{x}|\mathbf{T}_{0:k}, \mathbf{Z}_{0:k})$, and the multiple sampling on the entire camera trajectory questions its efficiency compared to modern keyframe-based SLAM systems.

Keyframe-Based SLAM: Modern SLAM systems starting from PTAM (Klein and Murray [2007a]) approach the computation differently. The system is divided into a front-end and back-end. The front-end deals with camera tracking and map estimation within a local window, and provide the map and pose state to the back-end as solution initialization. At the start neither the pose or the map is known, therefore the camera pose is initialized from two frames using the 5-point or 8-point algorithm (Nister [2003]), which is followed by alternation between map estimation for each *keyframe* given the camera pose(s) and camera pose estimation for each frame given the estimated map. Note that the choice of when to introduce a new keyframe is primarily based on amount of visible scene overlap.

In order to better understand how our work fits in to the SLAM front-end we can factor the joint posterior distribution that we intend to maximize as follows:

$$P(\mathbf{T}_{0:k}, \mathbf{x}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0) = P(\mathbf{x}|\mathbf{T}_{0:k}, \mathbf{Z}_{0:k})P(\mathbf{T}_{0:k}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0) \tag{2.10}$$

Since the camera poses are fixed during the map update, we essentially want to maximize $P(\mathbf{x}|\mathbf{T}_{0:k}, \mathbf{Z}_{0:k})$, and in this thesis we mainly focus on improving the model for $P(\mathbf{x}|\mathbf{T}_{0:k}, \mathbf{Z}_{0:k})$ for more accurate dense keyframe reconstruction, with the help of a CRF in order to model map priors (Section 2.2.3). After obtaining the map for the keyframe, the current camera state can be recovered by tracking the live image against the currently estimated map. The joint posterior can also be factored as follows:

$$P(\mathbf{T}_{0:k}, \mathbf{x}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_0) = P(\mathbf{T}_k|\mathbf{x}, \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_{0:k-1})P(\mathbf{x}, \mathbf{T}_{0:k-1}|\mathbf{Z}_{0:k}, \mathbf{T}_0) \tag{2.11}$$

Since the map is fixed during the current pose update, this means we want to maximize $P(\mathbf{T}_k|\mathbf{x}, \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_{0:k-1})$ during camera tracking. The current pose $\mathbf{T}_k$ is usually initialized based on the previous pose $\mathbf{T}_{k-1}$ using a motion model. In Chapter 4 we discuss how our learned features and associated learned confidences for matching can help improve the model for $P(\mathbf{T}_k|\mathbf{x}, \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{T}_{0:k-1})$ during camera tracking, through dense learned feature-based direct alignment (weighted by learned confidences). Another way to improve this model is to incorporate an image-dependent *learned* motion model if applicable as a prior on the current pose $\mathbf{T}_k$.

In modern SLAM systems such as ORB-SLAM (Mur-Artal et al. [2015]) keyframe-based bundle adjustment takes place in the back-end in a separate thread with initialization provided by the SLAM front-end. For practical reasons, the back-end processing is made more efficient via local bundle adjustment on the "covisibility graph" (formed by the set of keyframes in a local window) and pose-graph optimization (Strasdat et al. [2010b]) on the "essential graph" (formed by the whole set of keyframes minus some keyframes that have been pruned out)(Mur-Artal et al. [2015]) following the double window optimization strategy of Strasdat et al. [2011]. Other desired features such as place recognition (Cummins and Newman [2010]; Milford and Wyeth [2012]), and re-localization and loop closing are also taken care of typically in the back-end.

Some of the popular SLAM frameworks over the years are illustrated in Figure 2.3. We will go through some these frameworks briefly in the following subsections. Note that we will focus more attention on monocular visual SLAM methods. An advantage of the stereo camera scheme compared to the monocular one, is the property that 3-D features are computed directly in the absolute scale. Additionally, since the 3-D structure is computed directly from a single stereo pair rather than from adjacent frames as in the monocular case, the stereo scheme exhibits less drift than the monocular one in case of small motions. Monocular methods are interesting because stereo VO degenerates into the monocular case when the distance to the scene is much larger than the stereo baseline (i.e., the distance between the two cameras). In this case, stereo vision becomes ineffective and monocular methods must be used. Furthermore stereo reconstructions can also be improved further by making use of multiple redundant images captured over time by moving camera, regardless whether if its a monocular/stereo/multi-camera setup.

2.1.3 Online Sparse Methods

One of the first successful real-time monocular visual SLAM systems is MonoSLAM (Davison et al. [2007]). The underlying algorithm governing MonoSLAM is a Bayesian filter known as the Extended Kalman Filter (EKF). Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe. The extended Kalman filter (EKF) is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance.

EKF-based SLAM was previously introduced in Smith and Cheeseman [1986], however not performed on visual input. The work in Chiuso et al. [2002] was based on visual input but had the problem of not being able to close loops (perform re-localization) and did not bear many of the other practical benefits such as active mining of features in new frames present in Davison et al. [2007]. With the goal of mapping a larger number of landmarks, several variants of filter-based approaches were proposed. One such algorithm is by Eade and Drummond [2006] which uses the particle filter, based on the FastSLAM algorithm originally proposed in Montemerlo et al. [2002], along with sub-mapping techniques for scalability. Other variants also make use of the information filter for SLAM (Thrun et al. [2004]).

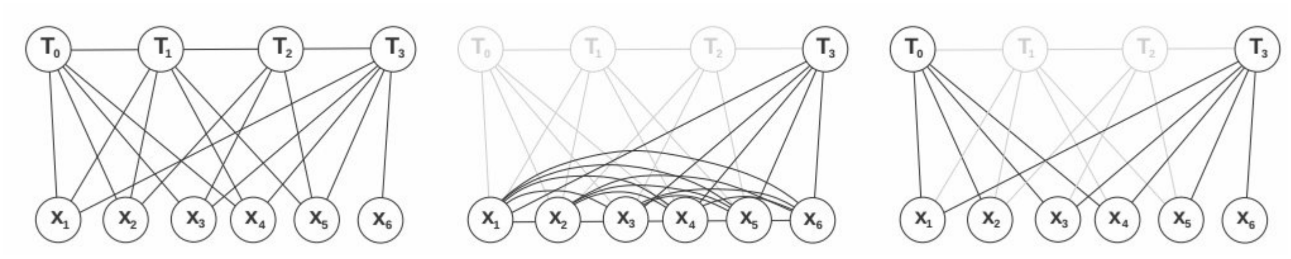


Figure 2.1: A high level abstraction of the Markov Random Fields illustrating the modeling complexity for (from left-to-right): Optimal BA (Bundle Adjustment) using all frames, Filter-based inference of map and current pose, and Keyframe-based BA (carried out in the back-end of modern SLAM systems). Notice how keyframe-based BA presents the most efficient graph structure for inference (for the same number of landmarks) by optimizing over keyframes, instead of marginalizing over all previous frames (filter-based) which results in the number of inter-dependencies that need to be modeled growing exponentially, or optimizing over all frames (optimal BA) which is wasteful considering landmark accuracy does not improve as much when triangulating based on small-baseline frames. The variables $\mathbf{T}_i, \forall i$ denote frame poses, and $\mathbf{x}_j, \forall j$ denote landmark locations. Modeled inter-dependencies are shown as dark edges in the graph which are conditioned on the observations (not shown). Figure adapted from Strasdat et al. [2010a].

The PTAM (Parallel Tracking and Mapping) (Klein and Murray [2007b]) algorithm presented a paradigm shift in sparse monocular SLAM and popularized the keyframe-based approach. The camera pose is tracked in real-time in one thread against the current *keyframe*'s map. In a separate thread, a sparse 3D map is estimated once a new keyframe is initialized based on the previous set of frames and their estimated poses — as previously discussed, this allows for tracking of more features for the same computational requirement as a filter-based method.

Strasdat et al. [2010b] proposed a method for optimizing 7-DoF camera poses based on similarity transformation defined on the "pose-graph" in order to correct the scale drift problem in monocular SLAM, and subsequently Strasdat et al. [2011] introduced a double window optimization scheme for constant-time visual SLAM. LSD-SLAM (Engel et al. [2014]) is able to build large scale semi-dense maps, using direct methods (i.e. optimization directly over image pixel intensities) instead of bundle adjustment over features. The system is able to operate in real time, without GPU acceleration, building a semi-dense map, with more potential applications for robotics than the sparse output generated by feature-based SLAM. Nevertheless they still need features for loop detection and their camera localization accuracy is significantly lower than ORB-SLAM (Mur-Artal et al. [2015]) and PTAM (Klein and Murray [2007a]).

The recent ORB-SLAM (Mur-Artal et al. [2015]) framework combines most of the positive aspects of the modern feature-based sparse SLAM algorithms, and is, at the time of writing, the state-of-the art and most complete open source framework for sparse feature-based localisation and mapping. Figure 2.2 depicts a summary of the ORB-SLAM pipeline, and we make extensive use of ORB-SLAM for camera tracking in our frameworks in Chapters 3 and 4.

SVO (Forster et al. [2014]) is a semi-direct visual odometry system. Although camera tracking is done by feature point matching, mapping is done via use of a direct method. Without requiring to extract features in every frame they are able to operate at high frame-rates obtaining impressive results in quadracopters. However no loop detection is performed and the current implementation is

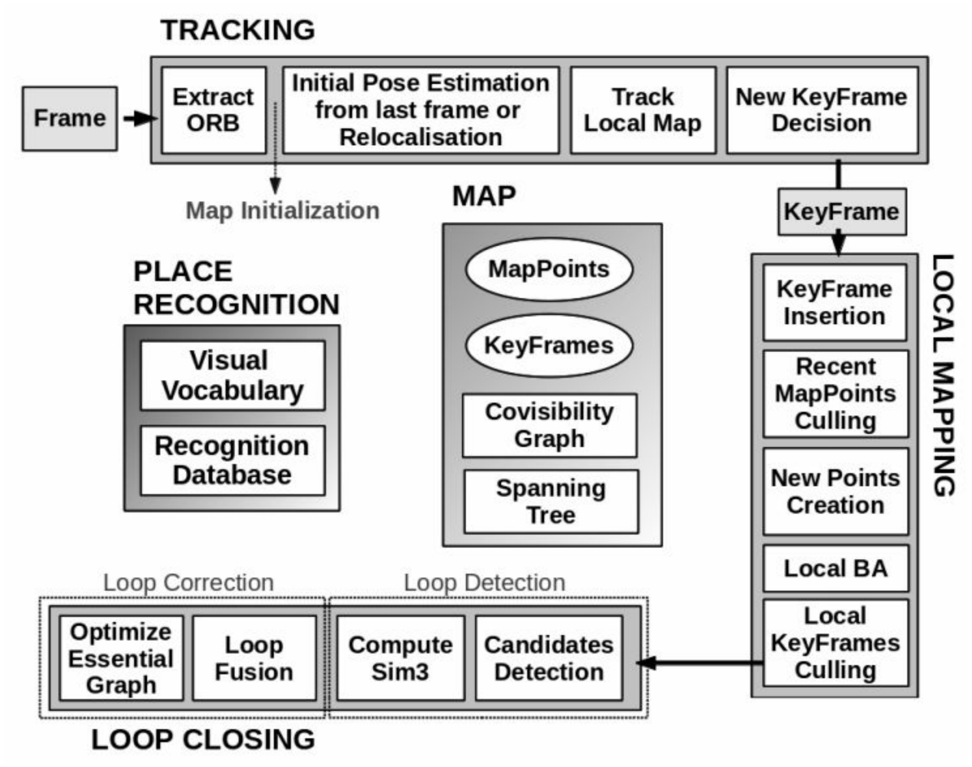


Figure 2.2: Illustration of the ORB-SLAM (Mur-Artal et al. [2015]) pipeline.

mainly thought for downward looking cameras. DSO (Direct Sparse Odometry)(Engel et al. [2017]) on the other hand is a fully direct visual odometry system and aims to combine the benefits or sparse and direct methods. The approach in DSO as well as ORB-SLAM are good candidates for the SLAM back-end due to computational efficiency reasons and complements our "just-in-time reconstruction" strategy for large scale mapping. With regard to the SLAM front-end, in Chapter 4 we take a step at combining the benefits of feature-based and direct methods for both *dense* mapping and tracking.

2.1.4 Online Dense Methods

Dense Visual SLAM: Newcombe et al. [2011b] proposed DTAM, a fully direct monocular dense SLAM system, which presented a paradigm shift in dense monocular SLAM. In DTAM, tracking is done via dense photometric alignment of the live image against the projection of the dense 3D map, and is efficiently implemented on the GPU with the help of a course-to-fine optimization strategy. Mapping is performed by creating a dense cost volume for the current keyframe, by accumulating frames over time and computing the cumulative matching cost against the current keyframe based on raw RGB values for multiple inverse depth hypothesis for each keyframe pixel. DTAM uses an inverse depth smoothness prior to fill parts of the scene with uniform texture that would otherwise not be possible to triangulate. Both tracking and mapping components exploit the massive parallel computational power of GPUs.

Stühmer et al. [2010] also proposed a variational approach for estimating the depth for every pixel, with the use of an inverse depth smoothness prior and a fully gradient based primal-dual optimization scheme (in contrast to the brute-force search coupled with primal-dual optimization scheme of DTAM

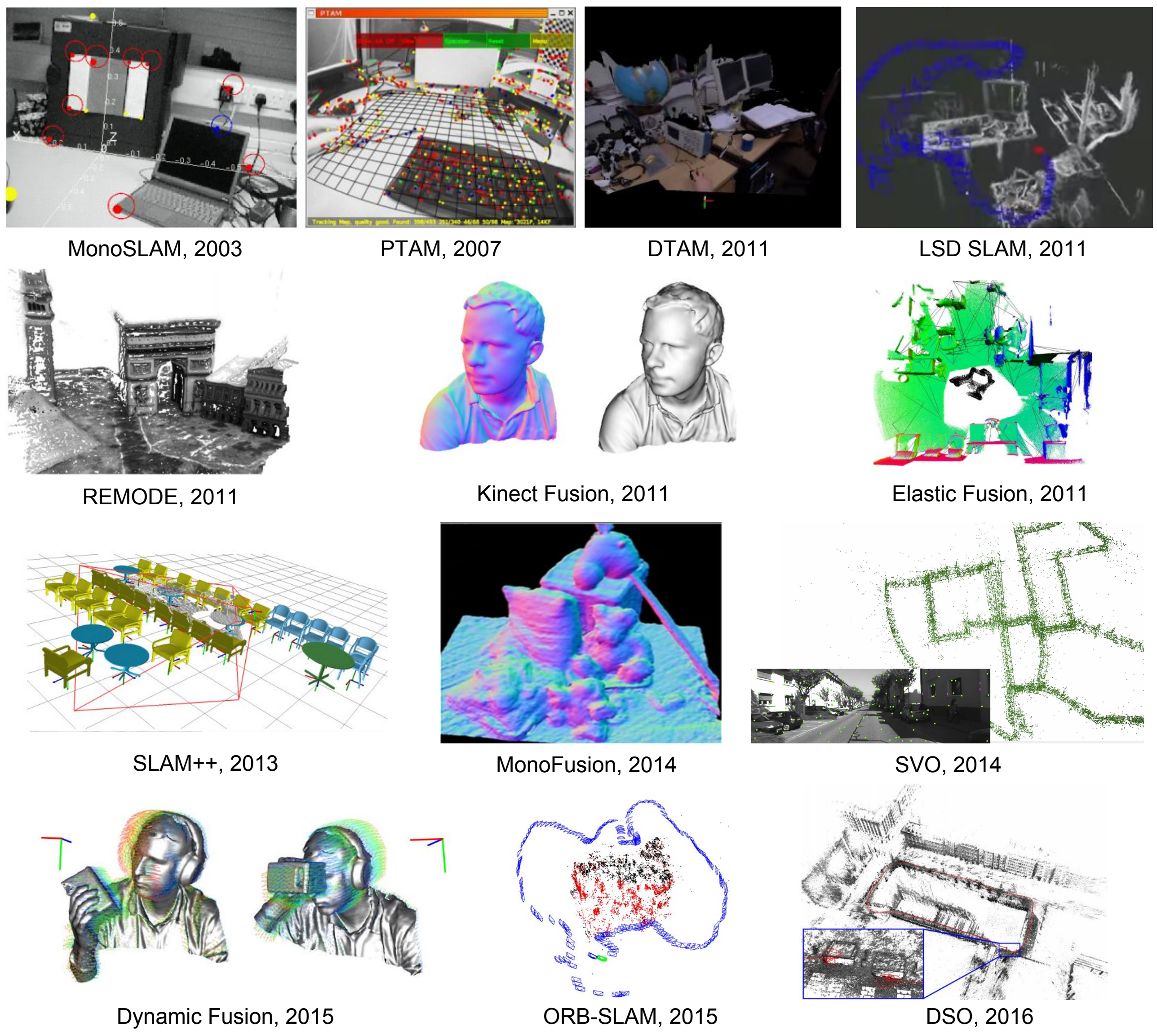


Figure 2.3: Illustration of some of the most popular visual SLAM systems over the years. Notice how most of the initial SLAM systems like MonoSLAM and PTAM were sparse and feature-based, followed by the first dense direct SLAM framework, DTAM. However recent focus once again has switched back to semi-dense and sparse SLAM like LSD-SLAM and ORB-SLAM. The figure also includes the popular RGB-D camera based Kinect Fusion, Elastic Fusion, and Dynamic Fusion dense SLAM systems, the latter of which is able to track and fuse 3D points in dynamic scenes.

that we employ in Chapters 3 and 4). They use a similar cost function for mapping as DTAM. In Stühmer et al. [2010], PTAM is used for tracking, and therefore is not a fully direct method like DTAM.

MonoFusion (Pradeep et al. [2013]) is a keyframe-based hybrid direct/feature-based method for monocular dense reconstruction. Here, dense stereo matching is done using current and past keyframes in order to infer depth maps. The live dense depths maps are fused into a voxel grid. The method does not require an explicit cost volume for stereo matching, and frames are directly fused into the voxel grid. A smoothness prior is applied on the depth map to regularize the depths. The 6-DoF (Degree of Freedom) camera pose is estimated using FAST corners (Rosten and Drummond [2006]) with planar patches, and using whole image alignment. It also supports camera relocalisation using tiny images.

In REMODE (Pizzoli et al. [2014]) the matching error distribution for each keyframe pixel as a function of inverse depth is approximated with a $Beta \times Gaussian$ distribution, and the product of such distributions is computed for a set of live frames. The same inverse depth regularizer as in DTAM is used for dense monocular reconstruction. While a performance comparison against the mapping framework of DTAM is lacking, in theory since REMODE does not perform optimization over the full cost volume as in DTAM (the latter of which doesn't approximate the matching cost function with a particular probability distribution), the performance of REMODE should be inferior (at least in the case where the features used for matching perform poorly and don't allow for a $Beta \times Gaussian$-distributed matching error function).

Comparatively less attention has been paid to the application of high-level scene understanding to aid the mapping process in dense visual SLAM. Some works have introduced constraints such as Manhattan world assumptions, or piecewise planar priors (Concha and Civera [2014, 2015]; Concha et al. [2014]; Flint et al. [2011]). Other stronger priors have also been leveraged, such as known objects (Bao and Savarese [2012]; Bao et al. [2013]; Dame et al. [2013]; Hane et al. [2013]; Kundu et al. [2014]; Ladický et al. [2012]).

Dense RGB-D SLAM: With the introduction of the Kinect sensor and readily available depth information about the scene, the task of 3D mapping is simplified. In Kinect Fusion (Newcombe et al. [2011a]), live sensory depth maps (which are partial and noisy) are fused in voxel space with the help of the TSDF representation (Figure 2.4). The camera motion is estimated by the Iterative Closest Point (ICP) algorithm which iteratively optimizes for the pose that aligns the live depth map with the fused volume based on point to plane distance.

It should be noted that most of the consumer depth cameras incorporating structured light sensor technology (essentially performing stereo in the IR domain) are developed for indoor usage. In addition fixed baseline stereo limits the range of depth measurements (although sufficient for most practical applications). Our framework in Chapter 5 aims at incorporating the best of all available depth sensing mechanisms through multi-model depth fusion.

Building on Kinect Fusion, Kähler et al. [2016] proposed a hierarchical TSDF representaiton and voxel hashing scheme. To reduce computational cost, they use voxel block hashing in the mapping process proposed originally in Nießner et al. [2013]. This improved implementation of Kinect Fusion is available via the open source InfiniTAM software (Prisacariu et al. [2014]) which we modify and use as a post-processing tool for fusing depth maps generated from our monocular SLAM framework in

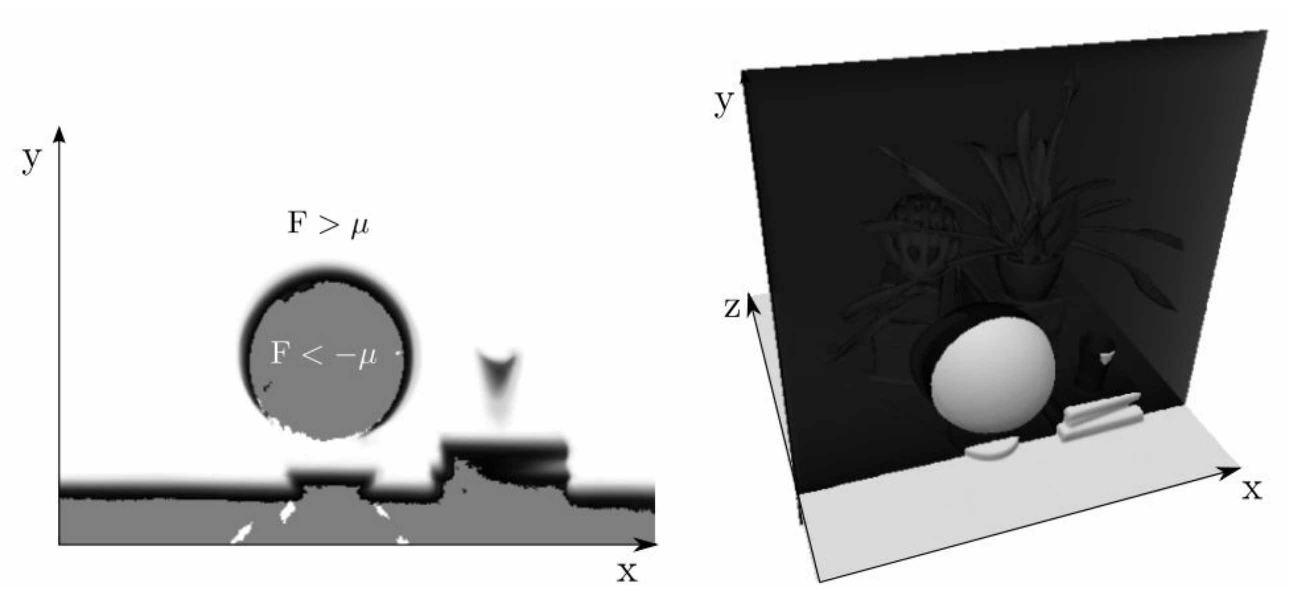


Figure 2.4: The Truncated Sign Distance Function (TSDF) for representing volumetric reconstructions. The figure shows a slice through the truncated signed distance volume showing the truncated function $F > \mu$ (white), the smooth distance field around the surface interface $F = 0$ and voxels that have not yet had a valid measurement (grey). Figure adapted from Newcombe et al. [2011a].

Chapters 3 and 4 of this thesis.

Elastic Fusion (Whelan et al. [2011]) produces surfel-based maps of room scale environments explored using an RGB-D camera in an incremental online fashion, without pose graph optimisation or any postprocessing steps. This is accomplished by using dense frame-to model camera tracking and windowed surfel-based fusion coupled with frequent model refinement through non-rigid surface deformations. Their approach applies local model-to-model surface loop closure optimizations as often as possible to stay close to the mode of the map distribution, while utilizing global loop closure to recover from arbitrary drift and maintain global consistency.

SLAM++ (Salas-Moreno et al. [2013]) is an object level RGB-D SLAM algorithm. In this method, several 3D objects are registered into a database in advance, and these objects are recognized in an online process. By recognizing 3D objects, the estimated map is refined, and 3D points are replaced by 3D objects to reduce the amount of data.

Dynamic Fusion (Newcombe et al. [2015]) showcased impressive dense 3D reconstruction results by extending Kinect Fusion to be able to fuse depth maps captured in dynamic scenes, with per-point 6-DoF (Degree of Freedom) tracking via point-to-plane-distance-based energy minimization coupled with a “as rigid as possible” prior.

2.1.5 Image Correspondence Estimation

Correspondence estimation across images is at the heart of all multi-view geometric reconstruction methods. In this section we discuss some commonly used features for measuring similarity of regions across images, particularly for dense every-pixel matching. Some of these features will be used as baselines in Chapter 4 for comparing against our *learned* “good features for matching”. We also briefly review some related work that learn features to match via Convolutional Neural Networks (CNNs). In

Chapter 4 we contrast our approach with the most related work in more detail.

RGB features: While low-level RGB features might seem insufficient for unique matching against other images, by accumulating evidence from a set of (usually a large number of) overlapping images, a surface reconstruction can be obtained by minimising the sum of photometric error across the images as in the case of DTAM (Newcombe et al. [2011b]):

$$E_\phi(\rho_p) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{I}_r(\mathbf{u}_p) - \mathbf{I}_n(\pi(T_{nr}\pi^{-1}(\mathbf{u}_p, \rho_p)))||_1 \tag{2.12}$$

E_ϕ is the dataterm that computes the photometric error for a keyframe $\mathbf{I}_r$ accumulated over N overlapping frames $\mathbf{I}_n$. $T_{nr} \in \mathbb{SE}(3)$ is a matrix describing the transformation of a point from camera coordinates of $\mathbf{I}_r$ to that of $\mathbf{I}_n$. $\pi(.)$ is the projection operation, and $\pi^{-1}(.,.)$ is the back-projection operation, such that $\pi^{-1}(\mathbf{u}_p, \rho_p) = K^{-1}\dot{\mathbf{u}}_p/\rho_p$, where K is the camera intrinsics matrix, and $\dot{\mathbf{u}}_p = (u, v, 1)^T$ is $\mathbf{u}_p$ (a pixel in the reference image) in homogeneous form.

RGB features are also used for camera tracking based on dense (Newcombe et al. [2011b]) or semi-dense (Engel et al. [2014]) image alignment. They are primarily used in direct SLAM methods where data association is performed implicitly through photometric error minimization. In Alismail et al. [2016a] raw RGB values are replaced with bit planes in order to robustify the matching process.

Image Patch Features: The most popular method for dense correspondence especially in stereo matching is template matching based on Sum of Squared Distances (SSD). This method is used for sparse matching in the DSO visual odometry system (Engel et al. [2017]). In the general case when given multiple overlapping images, we could compute the Sum of Sum of Squared Distances (SSSD):

$$SSSD(\mathbf{I}_r, \mathbf{I}_n, \rho_p) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{|\mathcal{N}_P(p)|} \sum_{q\in\mathcal{N}_P(p)} ||\mathbf{I}_r(\mathbf{u}_q) - \mathbf{I}_n(\pi(T_{nr}\pi^{-1}(\mathbf{u}_q, \rho_q)))||_2^2 \tag{2.13}$$

$\mathcal{N}_P(p)$ denotes the set of pixels in the patch centered at pixel p, including p. Note that the (inverse) depth of the center pixel (ρ_p) is typically used to represent the depth of the whole patch, resulting in an affine warped patch which is used for matching. A more sophisticated method would use for example estimated normals to interpolate the depths in a patch. In order to be resilient to global and local illumination changes, the image can be normalized locally or globally prior to matching. Another similarity measure employed for template matching that performs as well as SSSD is Zero-Normalized Cross-Correlation (ZNCC) where brightness normalized patches are compared using the correlation function. ZNCC was used for patch matching in MonoFusion (Pradeep et al. [2013]).

Handcrafted Feature Descriptors: Several hand-designed features, such as SIFT (Lowe [2004]), HOG (Dalal and Triggs [2005]), SURF (Bay et al. [2008]), ORB (Rublee et al. [2011]) and DAISY (Tola et al. [2010]) have found widespread applications. SIFT (Scale Invariant Feature Transform) is arguably one of the most popular among them and it is designed to be invariant to image rotations, affine transformations, intensity shifts, and viewpoint changes when matching. It is relatively costly to extract which inhibits its use in real-time applications, and led to the introduction of descriptors like SURF and ORB. The SIFT algorithm (Lowe [2004]) has 4 basic steps: (1) Estimate a scale space extrema using the Difference of Gaussian (DoG), (2) Perform keypoint localization based on image contrast and eliminate low contrast points, (3) Assign keypoint orientations based on local image gradients and (4) Generate local image descriptors for each keypoint based on image gradient magnitude and orientation. The Maximally Stable Extremal Region (MSER) blob detector (Matas et al. [2004]) is also commonly used as a means of finding which image regions to extract features. A dense version of SIFT (Dense SIFT) is used for dense correspondence tasks like optic flow (Liu et al. [2016a]) and involve keypoint sampling in a regular grid.

SURF (Bay et al. [2008]) approximates the DoG with box filters which are faster to compute and can be done in parallel for different scales. It uses a blob detector which is based on the Hessian matrix to find the points of interest. For orientation assignment, it uses Wavelet responses in both horizontal and vertical directions by applying adequate Gaussian weights and it also uses Wavelet responses for feature description. A neighborhood around a key point is selected and divided into sub-regions and the Wavelet responses are computed to form the SURF feature descriptor. The sign of the Laplacian (which is already computed during the detection step) distinguishes bright blobs on dark backgrounds and vice versa. Features are compared only if they have same type of contrast (based on sign) which allows for faster matching.

ORB (Oriented FAST and Rotated BRIEF) (Rublee et al. [2011]) is a fusion of the FAST keypoint detector (Rosten and Drummond [2006]) and BRIEF descriptor (Calonder et al. [2010]) with some modifications. The FAST detector is used initially to determine the keypoints. A Harris corner (Harris and Stephens [1988]) measure is then applied to find the top N points. FAST does not compute the feature orientation and is rotation variant. It computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to the centroid is therefore used to estimate the feature orientation and moments are computed to improve the rotation invariance. The BRIEF descriptor performs poorly if there is an in-plane rotation. In ORB, a rotation matrix is computed using the orientation of the patch and the BRIEF descriptors are steered according to this orientation.

CNN-Based Learned Feature Descriptors: Recently, some Convolutional Neural Network (CNN)-based similarity measures have been proposed. A Siamese network is used in Zagoruyko and Komodakis [2015] to measure patch similarity. A driving dataset is used to train a CNN for patch similarity in Agrawal et al. [2015], while Zbontar and LeCun [2015] also uses a Siamese network for measuring patch similarity for stereo matching. A CNN pre-trained on ImageNet is analyzed for visual and semantic correspondence in Long et al. [2014]. Correspondences are learned in Kanazawa et al. [2016] across both appearance and a global shape deformation by exploiting relationships in fine-grained datasets.

In a direction similar to ours, Choy et al. [2016] and Schmidt et al. [2017] propose to learn a metric space in which metric operations have direct interpretations about pixel similarity, rather than implicitly optimizing a CNN for patch similarity and using the intermediate features. They use a fully convolutional architecture that operates on the whole image, with a correspondence contrastive loss. Choy et al. [2016] additionally uses a convolutional spatial transformer for local patch normalization. Similarly, LIFT (Yi et al. [2016]) is a CNN-based feature descriptor but operates on image patches making it less efficient for dense matching. Neural networks are used in Bromley et al. [1994] for learning a mapping where the Euclidean distance in the space preserves semantic distance. The loss function for learning similarity metric using Siamese networks is subsequently formalized by Chopra et al. [2005]; Hadsell et al. [2006]. Recently, a triplet loss is used by Wang et al. [2014] for fine-grained image ranking, while the triplet loss is also used for face recognition and clustering in Schroff et al. [2015]. Mini-batches are used for efficiently training the network in Oh Song et al. [2016].

2.2 Scene Understanding for Geometry Estimation

In this section we focus on methods that estimate geometry based on visual features, mainly those that require just a single image. We first introduce the Convolutional Neural Network (CNN) which is the primary tool we employ for estimation of geometry (surface normals and depth maps) and associated confidences from a single image. We also introduce the Conditional Random Field (CRF) probabilistic graphical model which is a powerful tool that we use extensively to model and probabilistically fuse geometry estimates from multi-view geometry/sensory data with geometry estimates via scene understanding.

2.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep, feed-forward artificial neural networks. In contrast to multi-layer perceptrons that have fully connected layers (and therefore dimensionality of model grows exponentially with size of input making them harder to train), CNNs have the distinctive advantage of having *local receptive fields*, and *shared weights* (or weight replication) and have been very successful (even more so recently) for a wide variety of visual recognition tasks. CNNs have been around since the 1980s and one of the pioneering architectures that most modern architectures are built on is LeNet-5, a 7 stage CNN by Lecun et al. [1998] (Figure 2.5). The idea of connecting units to local receptive fields on the input goes back to the introduction of the Perceptron in the early 60s, and the discovery of locally-sensitive, orientation-selective neurons in the cat's visual visual system (Hubel and Wiesel [1962]).

The typical training objective of a neural network is to iteratively update the model weights W as to minimise the average loss defined over all $|D|$ data instances throughout the dataset:

$$L(W) = \frac{1}{|D|}\sum_{i}^{|D|} f_W(X^{(i)}) + \lambda_r r(W) \tag{2.14}$$

where $f_W(X^{(i)})$ is the loss on the data instance $X^{(i)}$ and $r(W)$ is a regularization term (typically L1 or L2 norm of W) with weight λ_r. In practice $|D|$ can be very large so in each weight update iteration, a stochastic approximation of this objective is used, drawing a mini-batch of $N << |D|$ instances.

$$L(W) \approx \frac{1}{N}\sum_{i}^{N} f_W(X^{(i)}) + \lambda_r r(W) \tag{2.15}$$

The neural network model computes f_W in the forward pass which involves passing the input $X(i)$ through several network layers. The gradient ∇f_W is computed in the backward pass and involves

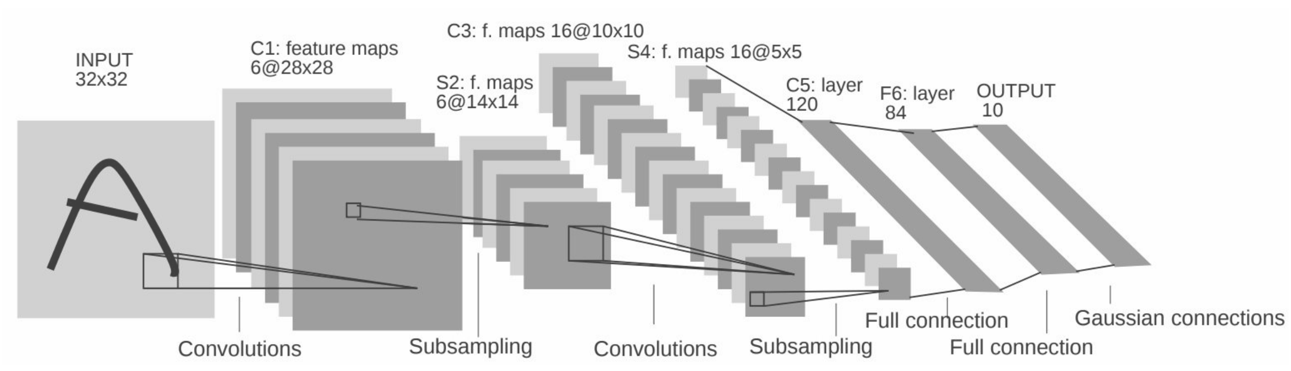


Figure 2.5: The architecture LeNet-5 (Lecun et al. [1998]), of one of the successful pioneering CNNs applied for hand written character recognition.

application of the chain-rule where each layer in the chain computes the gradient of its output with respect to its input, so that ∇f_W at each layer in the chain is a product of these individual gradients, starting from the final loss layer and working backwards until one layer up — a process known as "backpropagation" — times the gradient of the output of that layer with respect to its weights. The parameter update ∇W is formed by the solver from the error gradient ∇f_W, the regularization gradient $\nabla r(W)$, and other particulars to each method. Stochastic Gradient Descent (SGD) is the most commonly used optimization method. It updates the weights W by a linear combination of the negative gradient ∇L and the previous weight update V_t. The learning rate α is the weight of the negative gradient. The momentum μ is the weight of the previous update. Formally, we have the following formulas to compute the update value V_{t+1} and the updated weights W_{t+1} at iteration $t+1$, given the previous weight update V_t and current weights W_t (Bottou [2012]):

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t) \tag{2.16}$$

$$W_{t+1} = W_t + V_{t+1} \tag{2.17}$$

Other optimisation methods in the literature include AdaDelta (Zeiler [2012]), AdaGrad (Duchi et al. [2011]), Adam (Kingma and Ba [2014]), Nesterov's Accelerated Gradient (NAG) (Sutskever et al. [2013]), and RMSprop (Hinton et al.).

The following layers are the main constituents of CNNs:

Convolution Layer: The convolution layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. *3D Volumes of Neurons:* The layers of a CNN have neurons arranged in 3 dimensions: width, height and depth. The neurons inside a layer are connected to only a small region of the layer before it, called a receptive field. Distinct types of layers, both locally and completely connected, are stacked to form a

CNN architecture.
Local Connectivity: Following the concept of receptive fields, CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers. The architecture thus ensures that the learnt "filters" produce the strongest response to a spatially local input pattern. Stacking many such layers leads to non-linear filters that become increasingly global (i.e. responsive to a larger region of pixel space) so that the network first creates representations of small parts of the input, then from them assembles representations of larger areas.
Shared Weights: In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map. This means that all the neurons in a given convolutional layer respond to the same feature within their specific response field. Replicating units in this way allows for features to be detected regardless of their position in the visual field, thus constituting the property of translation invariance.

Together, these properties allow CNNs to achieve better generalization on vision problems. Weight sharing dramatically reduces the number of free parameters learned, thus lowering the memory requirements for running the network and allowing the training of larger, more powerful networks.

Deconvolution Layer: The deconvolution layer also called transposed convolution layer or fractionally strided convolutional layer performs an operation that is equal to the one used for backpropagation in the convolution layer. During backpropagation in the convolution layer we distribute a *weighted* gradient (based on the weights of the convolutional kernel) from each node in the output feature map to all the nodes in the input feature map corresponding to the receptive field of each output node, and weighted gradients from multiple output nodes are summed up at each input node. The same operation happens in the deconvolution layer except that the gradients of the output nodes in the convolution layer now represent the input values to the deconvolution layer and the input feature map to the convolution layer now represent the output feature map of the deconvolution layer. The deconvolution layer is primarily used as a learnable upsampler (as in our proposed neural network in Chapter 4) and in the decoder portion of a convolutional auto-encoder neural network.

Pooling Layer: Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance.

ReLU Layer: The ReLU (Rectified Linear Unit) layer applies the non-saturating activation function $f(x) = \max(0, x)$. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent $f(x) = \tanh(x)$, and the sigmoid function $f(x) = (1 + e^{-x})^{-1}$. ReLU is often preferred to other functions, because it trains the neural network several times faster (Krizhevsky et al. [2012]) without a significant penalty to generalisation accuracy.

Fully Connected Layer: Finally, after several convolutional and max pooling layers, the high-level reasoning (or transfer of information from one modality to another) in the neural network is typically done via fully connected layers (although convolutional layers with high number of output filters and large receptive fields are also capable of the same). Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. A fully connected layer can be viewed as a special case of a convolutional layer where the kernel/filter size is same as the spatial dimensions of the input blob to the layer. Their activations can be computed with a matrix multiplication followed by a bias offset.

Loss layer: The loss layer specifies how training penalizes the deviation between the predicted and true labels and is normally the final layer. Various loss functions appropriate for different tasks may be used there. The Euclidean loss (sum of squared L2 losses) is typically used for regressing to real-valued labels, like depth values. When the output of the neural network is a discrete set of classes the total loss is typically computed by taking the average of all cross-entropies in the sample. For example, suppose we have N classes in a sample indexed by $n = 1, \dots, N$. Let us define the true probability for a class in the sample as $p \in \{p, 1-p\}$ and the estimated probability as $q \in \{\hat{p}, 1-\hat{p}\}$ which is a function of the neural network weights $\mathbf{w}$. The cross-entropy (or log) loss is defined as:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}\Big[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\Big] \tag{2.18}$$

If the output classes are independent, then the probability $\hat{y}_n$ can come from a sigmoid or logistic activation function. In this case, the overall loss is known as binary/multi-label cross entropy loss, or sigmoid cross entropy loss or simply logistic loss. If the output classes are mutually exclusive, the probability $\hat{y}_n$ should come from a probability distribution, and thus can come from the softmax function, $f(x_n) = e^{x_n} / \sum_i e^{x_i}$. In this case, the overall loss is called the multi-class cross-entropy loss or softmax cross entropy loss. We employ the latter loss function for our feature learning method in Chapter 4.

2.2.2 Convolutional Neural Networks for Geometric Scene Understanding

CNNs for Single/Multi-View Depth, Normal, Optic Flow, and Camera Pose Estimation: Depth estimation from a monocular color image is a long-standing problem in computer vision. Early work mainly aimed at estimating surface normals from a single image and in turn integrated them to form depth maps (Durou et al. [2016]). Normals were estimated using shape from shading (Zhang et al. [1999]), shape-from-defocus (Suwajanakorn et al. [2015]) and other low-level image features (Hoiem et al. [2005, 2007]). In Criminisi et al. [1999] 3D distances between points in an image were inferred based on vanishing points and lines.

In Brooks and Horn [1985], the iterative scheme for computing shape from shading adjusts the current estimates of the focal normals toward or away from the direction of the light source. The amount of adjustment is proportional to the current difference between the predicted and the observed brightness. In contrast, learning based approaches such as Fouhey et al. [2013]; Ladický et al. [2014], especially those using neural nets (Bansal et al. [2016a,b]; Eigen and Fergus [2015]; Wang et al. [2015]) have achieved state-of-the art performance in surface normal prediction.

One of the early learning based methods to achieve reasonable success in single image depth estimation is Make3D (Saxena et al. [2006, 2009]) which relied upon a set of handcrafted features to map patches from the input image to some feature space, and in turn learn to regress from that feature space to depth values. They also use an additional pairwise depth smoothness prior, modeled as either a Gaussian or Laplacian distribution with a data dependent learned variance, in order to regularize and globally optimize for a depth map, with the belief that the process of recovering a depth map requires global reasoning on the image.

Recently, deep learning-based methods dominate this area (Eigen et al. [2014]; Kendall and Gal [2017]; Xie et al. [2016]). For example, Eigen et al. [2014] train a multi-scale Convolutional Neural Network, operating at coarse and fine image resolutions, to regress a depth map from a single image, and in Eigen and Fergus [2015] they extend their network to a three-scale architecture and regress for depth maps, normal maps, and semantic labels in real-time from a single image. The semantic label maps were predicted from a single RGB-D image as the additional depth channel improved results. The depth and normal network consists of shared weights, while the semantic segmentation network consists of separate weights. All three tasks however share the same architectural design. In Dharmasiri et al. [2017] the latter work was extended to jointly predict depth, surface normals and surface curvature, which improved the results of all three tasks.

Liu et al. [2016b] proposed to formulate depth estimation as a deep continuous Conditional Random Fields (CRF) learning problem. Given the continuous nature of the depth values, they learn the unary depth values and weightings for the pairwise potential functions via CNNs in an end-to-end framework. Laina et al. [2016] used a fully convolutional network architecture based on ResNet (He et al. [2016]) with a novel upsampler for decoding the depth map at input resolution. Roy and Todorovic [2016] combined shallow convolutional networks with regression forests to reduce the need for large training sets. Recently in Hu et al. [2018], it was proposed that sharper predictions at depth boundaries can be achieved by emphasizing local depth error gradients. This same phenomenon was observed in Eigen and Fergus [2015]; Ummenhofer et al. [2017] that also emphasized local depth errors during training. Note that all of these methods are trained in a supervised manner to reproduce the raw depth acquired with commodity RGB-D cameras.

Garg et al. [2016] presented a novel framework for depth estimation that can be trained with an unsupervised loss. The approach uses pairs of images to construct a photometric error loss, and a CNN is trained to minimize this loss (and satisfy an inverse depth smoothness prior) by regressing for the optimal inverse depth values from a single image that parameterize the image warp function. Godard et al. [2017] extended this loss function by incorporating a left-right depth consistency loss as well as a SSIM (structural similarity) loss to obtain improved depth prediction results. A series of work followed upon this idea (Vijayanarasimhan et al. [2017]; Wang et al. [2017]; Yang et al. [2017]; Zhan et al. [2018];

Zhou et al. [2017]) with slightly improved methods and results and some of them also learn to predict egomotion (from two views) in an unsupervised manner based on the same unsupervised loss and stereo and/or monocular training image data.

In Hane et al. [2015], a similar method as in Hane et al. [2014] was used for refining single-image depth predictions, or stereo-based depth estimates using surface normal predictions from a neural network. Pixel-wise classification scores of a discrete set of learned surface normals were used to pre-compute a Wulff shape for each pixel in the keyframe which in turn determined the amount of directional smoothing of the reconstructed surface. We contrast our formulation to theirs in Chapter 3. In a parallel line of work to ours, SURGE (Wang et al. [2016]) also used surface normals predicted from a neural network to refine single view depth estimates. The formulation is similar to ours in Chapter 3, in that an energy term was proposed to enforce consistency between pixelwise surface normals and difference vectors in 3D, some minor differences being that their formulation used depth parameterization, and applied a quadratic penalty on the pairwise terms which formed a fully connected graph.

The recent CNN-SLAM (Tateno et al. [2017]) system, fused single-view depth predictions from a CNN across keyframes together with depth from multi-view stereo via simple weighted averaging. They use multi-view depth consistency as a means to obtain confidence weights for the CNN depths. Some noteworthy differences between their fusion approach and ours in Chapter 5 are as follows. Their method places learned depth priors on the nodes of a CRF. In contrast our fusion method places learned depth priors on the edges of a CRF, which makes our prior more robust to errors in absolute depth values, given that absolute depth is often more difficult to learn (Eigen et al. [2014]). Also, we obtain depth confidences (proportional to the inverse of the depth variance) via *prediction* from a single image and depth map while the approach of CNN-SLAM to obtain depth confidence is more heuristic-based and require depth estimates from multiple views.

In Cadena et al. a multi-model autoencoder is used to create a shared code which represented an image, its depth map and semantic label map. In CodeSLAM (Bloesch et al. [2018]) a single view depth prediction network is concatenated with a depth map auto-encoder so that the latter can learn complementary information and the overall network can jointly produce improved depth estimates than what a single RGB image-only based depth predictor could produce. This is an interesting direction for efficient large scale dense visual SLAM systems, as a compact code (conditioned on an RGB image and fixed neural network weights) can serve as a compact representation for a dense depth map. When provided with multiple images overlooking a scene from different views, only the code needs to be refined such that the decoded depth map satisfies multi-view photo-consistency. The image-guided depth decoder also acts as an implicit depth regularizer, limiting the space of depth maps to a plausible set during code optimization. CodeSLAM is leaning towards the ideas advocated in this thesis and in Chapter 6 we examine how it can complement the work we present. Similar to CodeSLAM, in Tang and Tan [2018] a code is passed through an image-guided learned decoder, and learned feature error is used (instead of photometric error) as the loss function inside the Levenberg–Marquardt algorithm for code optimisation.

FlowNet (Dosovitskiy et al. [2015]) presented an end-to-end network for dense optic flow estimation from two images. Recently, in Dharmasiri et al. [2018] a neural network is also trained for dense optic flow estimation from two images, which in turn is subsequently used for pose estimation via dense alignment. While impressive results in pose estimation are observed it is interesting to answer the question if optic flow prediction is necessary for mapping and camera tracking when having "good features to match" (Chapter 4). It might be that for structured ego-motion estimation such as in the case a camera is mounted on a moving vehicle, optic flow/direct ego-motion predictions from images can serve as a useful prior, for instance, in the motion model for camera tracking. However general hand-held camera movements present no (or very little) motion patterns that can be learned, and it is perhaps more generalizable to solely use a direct method that exploits traditional epipolar geometry and "good features to match" to address the correspondence problem (especially in the rigid scene case).

In DeMoN (Ummenhofer et al. [2017]), 6-DoF egomotion, optic flow and a depth map are all estimated from two images in an end-to-end manner via a CNN. Kendall and Cipolla [2017] train a network for end-to-end regression of 6-DoF camera pose given a single image for the (re-)localization task (for scenes that are in the training set) and Valada et al. [2018] show that the visual localization and two-frame visual odometry tasks can be solved jointly within one network. In Kendall et al. [2017] a network was trained end-to-end to produce state-of-the-art disparity estimations (from stereo images) on the KITTI dataset. Very recently, in DeepTAM (Zhou et al. [2018]), images captured at two or more views are used to construct a cost volume similar to DTAM which in turn is fed as input to a neural network for depth map prediction. They also train a separate neural network for pose estimation from two views and a depth map, as an end-to-end learning based replacement for dense tracking. In a future work, it would be interesting to compare the performance and generalizability (performance outside the training set domain) of the very recent DeepTAM's purely learning-based end-to-end two frame/multi-frame approach for depth and pose estimation against our approach in Chapters 3 and 4, which is a mix of learning-based and traditional geometry-based methods.

CNNs for Depth Inpainting: Recently, methods have been proposed for inpainting color images with auto-encoders (Oord et al. [2016]), GANs (Generative Adversarial Network) (Pathak et al. [2016]), and partial convolutions (Liu et al. [2018]). However, prior work has not investigated the use of such methods for in-painting sparse depth maps, given only a sparse depth map at the input to a network. This problem is more difficult due to the absence of strong features in depth maps alone.

In Ma and Karaman [2017] an end-to-end framework for densifying sparse depth maps was proposed where a randomly sampled Kinect/LIDAR depth map was concatenated with a RGB image at the input to a network. This network too was trained to implicitly fuse the information in feature space and regress for a dense depth map. They showed that providing ≈ 100 well-spaced depth samples improves depth estimation over color-only methods by two-fold on the NYU-D v2 dataset, yet still with relatively low-quality results.

The above work differs to ours, particularly that in Chapter 5, as we explicitly fuse depth information predicted from the RGB image with a partial/noisy depth map in a probabilistic manner via optimization. During our initial experiments for "just-in-time reconstruction", we followed a similar strategy as Ma and Karaman [2017]. However, our proposed approach produce superior results due to the explicit probabilistic modeling and global inference for ensuring consistency with the sparse observations. In addition to this, in our approach we do not have to retrain the CNN depth network to suit the sparsity structure and noise of the input depth map regardless of the source.

In a parallel line of work to ours, Zhang and Funkhouser [2018] used learned normals for RGB-D inpainting. The motivation of their work is very similar to ours in Chapter 5, and their normal-based prior formulation is similar to ours presented in Chapter 3 (although their formulation imposes a constraint in depth space rather than in inverse depth space as ours). Their experiments showed that explicit data fusion via global optimization provide better results than allowing the network to fuse the RGB and sparse depth data implicitly. Moreover they compare surface normal and depth map predictions from RGB-D input (with partial depth) with those from RGB input alone, and found that learned normals and depths from RGB input alone were more accurate at regions with no depth at input. The intuition behind the result is that, in the RGB-only input case, the network was forced to learn the more complex relationship between the image and depth map, rather than the easier task of inpainting the partial depth map in the RGB-D input case. This led to more accurate results specifically in large regions in the output depth map with no corresponding depths at the input. Their experiments further support our proposed frameworks in Chapters 3 and 5 where normals/depths are predicted based on the RGB image alone, and subsequently refined via global optimization in order to satisfy the multi-view geometry-based or sensory observations.

CNNs for Uncertainty Estimation: Predicting uncertainty from data is a reliable and fast way to capture underlying limitations and uncertainty of the models estimating depth (e.g. CNN depth model, SLAM model, etc.) as seen from our experiments in Chapter 5 and in Kendall and Gal [2017]. However this method comes with its limitations. During confidence learning, it is important to account for the performance of the depth estimation models when given input outside the training set, in order to more accurately capture model uncertainty in such cases. This is more important for the CNN depth prediction model that generally perform worse outside the training set. This means we need to train on a large number of different datasets in order to correctly capture the CNN depth model uncertainty. Note that the accuracy of confidence prediction will still be limited by the capacity of the confidence prediction model itself. A neural network predicting uncertainty from data (especially one that is trained on limited data) is better at capturing aleatoric uncertainty of the model, or the "known unknowns" seen at train time (Kendall and Gal [2017]).

The Monte Carlo dropout method, computing uncertainty based on random sampling of a CNN output at test time (Gal and Ghahramani [2016]) is empirically shown in Kendall and Gal [2017] to be better at capturing CNN model uncertainty for input rarely seen during training, i.e. epistermic uncertainty, or the "unknown unknowns". In this sampling-based approach we relate variability of the output to variance. However this approach has a drawback as well. There are certain observations for which the depth estimation model can *consistently* produce wrong output as it exceeds its modeling capacity, causing systematic errors. For SLAM, observations such as those from specular surfaces lead to this behavior. The reconstructed output for such observations consistently satisfy the SLAM geometric model which is built on the assumption of scene rigidity.

For the CNN depth model, observations that cause systematic errors may include image edges and large textureless regions with little information about vanishing lines and points. Image edges are a problem for CNN depth estimators as fine details are typically lost as data flows through the network, due to progressive downsampling and receptive field increase. Large textureless regions could be a problem due to lack of implicit global reasoning in the CNN depth estimator. Indeed, the finite modeling capacity of CNNs adds to the problem. Therefore for such observations the CNN depth estimator can consistently produce the wrong result when randomly sampled leading to over-estimation of confidence (Kendall and Gal [2017]).

For this reason, the sampling method is best used in combination with the uncertainty prediction method as empirically shown in Kendall and Gal [2017]. Apart from this, the dropout sampling-based approach inhibits real-time performance (e.g. 50x slowdown for obtaining 50 output samples in the DenseNet architecture in Kendall and Gal [2017] bottlenecked by GPU memory and throughput). As a minor consideration, one should train the CNN depth predictor with dropout-enabled layers for this uncertainty estimation method.

In this thesis, due to focus on efficiency, we utilize the former method of predicting uncertainty directly from the data.

2.2.3 Conditional Random Fields (CRFs)

A Conditional Random Field (CRF) is a type of discriminative undirected probabilistic graphical model (Koller and Friedman [2009]). It is often used for labeling or parsing of sequential data. In computer vision, CRFs are useful for tasks like depth map estimation and semantic segmentation, as they can model relationships of the hidden variables corresponding to the image pixels to produce consistent interpretations of the obsersations.

In contrast to Markov Random Fields (MRFs), which are primarily generative models, CRFs model the conditional probability distribution, rather than the joint probability distribution, and are primarily discriminative models. CRFs are typically easier to model, as modeling the joint distribution is not straightforward, and also not necessary for discriminative tasks.

More specifically, constructing a CRF involves modeling the following conditional probability distribution:

$$P(\mathbf{y}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{y})P(\mathbf{y})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\mathbf{y})P(\mathbf{y})}{\int_{\mathbf{y}} P(\mathbf{x}|\mathbf{y})P(\mathbf{y})d\mathbf{y}} = \frac{exp(-E(\mathbf{y}))}{\int_{\mathbf{y}} exp(-E(\mathbf{y}))d\mathbf{y}} \tag{2.19}$$

$\int_{\mathbf{y}} exp(-E(\mathbf{y}))d\mathbf{y} = Z$ is called the partition function. Taking the negative log likelihood of (2.19) we obtain the following energy functional:

$$E(\mathbf{y}) = -log(P(\mathbf{y}|\mathbf{x})) = E_{data}(\mathbf{y}) + E_{reg}(\mathbf{y}) - lnZ \tag{2.20}$$

It is quite common in computer vision to model a problem as inference of an energy function consisting of a data term E_{data} and regularization (or prior) term E_{reg}, corresponding to the observation model $P(\mathbf{x}|\mathbf{y})$ and prior model $P(\mathbf{y})$ respectively. The negative log partition function is usually ignored during inference as it is independent of (marginalized over) $\mathbf{y}$. However this normalization function becomes important when learning the model parameters such as variances (as we see in Chapters 3, 4, and 5).

$P(\mathbf{y})$ is generally approximated by some assumption (prior) regarding interdependencies of scene

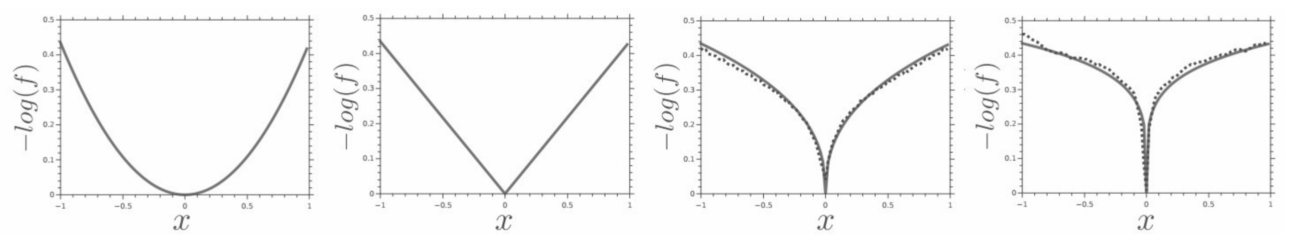


Figure 2.6: Plots for the negative likelihood of four instances of the generalised Gaussian distribution. From Left to Right: (1) Gaussian ($\alpha = 2$) and (2) Laplacian ($\alpha = 1$) distributions, (3) the generalised Gaussian model fit $\alpha = 0.8$ (solid line) for the -log of sampled probability distributions (dashed) for the gradients from a grey-scale image ∇I captured from a moving camera, and (4) similarly the Gaussian model fit $\alpha = 0.2$ (solid line) for the -log of sampled probability distributions (dashed) for the corresponding depth image gradient $\nabla \mathbf{y}$ captured from the same camera trajectory taken over the course of a minute of scene browsing in an office environment. Figure adapted from Newcombe [2012].

depth values, and in the case of DTAM, $E_{reg}(\mathbf{y}) = -ln(P(\mathbf{y})) = |\nabla \mathbf{y}|$, i.e. a prior formulated based on the assumption that the inverse depth values vary smoothly. The optimal solution $\mathbf{y}^*$ can be found by minimising $E(\mathbf{y})$, which is equivalent to obtaining the maximimum a posterior estimate for $\mathbf{y}$ that maximizes $P(\mathbf{y}|\mathbf{x})$.

In the case of DTAM, a question aries in what probability distribution we should choose to model the inverse depth smoothess prior. In Newcombe [2012] it was illustrated emperically that the distribution of the depth map gradient as well as the image gradient for natural images follow a generalised Gaussian distribution with higher kurtosis and larger variance than is achieved with a Gaussian distribution:

$$f(x) \propto exp(-\frac{|x-\mu|^{\alpha}}{\alpha\sigma^{\alpha}}) \tag{2.21}$$

As shown in Figure 2.6, taking the negative log likelihood of the computed histogram over depth map gradients $\nabla \mathbf{y}$ shows clearly that a non-convex penalty over the first order gradients is appropriate, with a good fit to the generalised Laplace distribution at $\alpha = 0.2$ (Newcombe [2012]).

However, since a convex formulation allows for fast global optimisation, an interesting prior model arises at the boundary between convex and non-convex where $\alpha = 1$, the closest convex model to the desired distribution, the Laplace distribution. The equivalent penalty function is the L1 norm.

In comparison to optimisation under the quadratic penalisation, the L1 norm presents a robust cost which when applied as prior with the first order derivatives of the solution yields Total-Variation (TV) regularisation of the solution. In signal processing, total variation denoising, also known as total variation regularization, is a process, most often used in digital image processing, that has applications in noise removal. It is based on the principle that signals with excessive and possibly spurious detail have high total variation, that is, the integral of the absolute gradient of the signal is high. According to this principle, reducing the total variation of the signal subject to it being a close match to the original signal, removes unwanted detail whilst preserving important details such as edges. The concept was pioneered by Rudin, Osher, and Fatemi and so is known as the ROF model (Rudin et al. [1992]).

This noise removal technique has advantages over simple techniques such as linear smoothing or median filtering which reduce noise but at the same time smooth away edges to a greater or lesser degree. By contrast, total variation denoising is remarkably effective at simultaneously preserving edges whilst smoothing away noise in flat regions, even at low signal-to-noise ratios (Strong and Chan [2003]).

It is important to note however that the (inverse) depth smoothness prior yields a distribution with high variance and therefore is less informative. *In this thesis, by replacing the (inverse) depth smoothness prior with a pairwise (inverse) depth prior based on learned geometry from an image, we essentially make the error distribution closer to a Gaussian with lower variance, resulting in a stronger and more useful prior for dense reconstruction, and at the same time making the choice of L1 norm or even quadratic penalty more reasonable.*

An alternative and more realistic depth prior (in comparison to the first order (inverse) depth smoothness prior) is the second order (inverse) depth smoothness prior, i.e. smoothness in (inverse) depth gradient. This prior yields Total Generalized Variance (TGV) regularization of the solution. Through experimentation, in Newcombe [2012] it was found that while the TGV solution produces the most complete solutions at low error thresholds, that it generally produces less accurate reconstructions in practice with errors resolving at depth discontinuities.

It was concluded that this is possibly due to the higher order smoothness term trading off immediate jumps at such boundaries possible with TV against second order smoothness that can further reduce error in larger planar regions. This means that TGV regularization produces lower bandwidth (piecewise affine) reconstructions due to the increased derivative filter size. It was also noted in Newcombe [2012] that the TGV model requires up to 10$\times$ more iterations than the first order models to converge to a useful state.

This raises an important issue when attempting to evaluate components of a live dense reconstruction system: given a fixed window of available processing time, a trade-off exists between attempting to compute the highest quality solution over a subset of frames and computing a lower quality solution over a larger set of frames (Newcombe [2012]). For these reasons the first order (inverse) depth gradient is more commonly employed in the literature, and in Chapters 3 and 4, we will be employing this first order (inverse) depth gradient prior (as in DTAM) as our baseline.

2.3 Legendre-Fenchel Transform

The Legrendre-Fenchel Transform finds the convex conjugate $f^*(q)$ of a *continuous* (but not necessarily convex or continuously differentiable) function $f(x)$, facilitating the use of efficient convex optimization methods. The conjugate is parametrized by the dual variable q. More formally this transformation is defined as:

$$f^*(q) = \sup_x \langle q, x\rangle - f(x) \tag{2.22}$$

The transform is motivated by the fact that there are two ways of viewing a curve or a surface, either as a locus of points or envelope of tangents (Rockafellar [1997]). A tangent at a point x_0 can be parameterised by two variables namely the slope $q = \dot{f}(x_0)$ and the intercept it cuts on negative y-axis, $c = \langle q, x_0\rangle - f(x_0)$. For each slope q, the goal is to find x_0 that gives the largest negative y-intercept c, in which case $c = f^*$, i.e. the convex conjugate of f (Figure 2.7). The pairs of $(q, f^*), \forall q$ form the envelope of tangents and the pairs of $(x, f), \forall x$ form the locus of points that describe the function f.

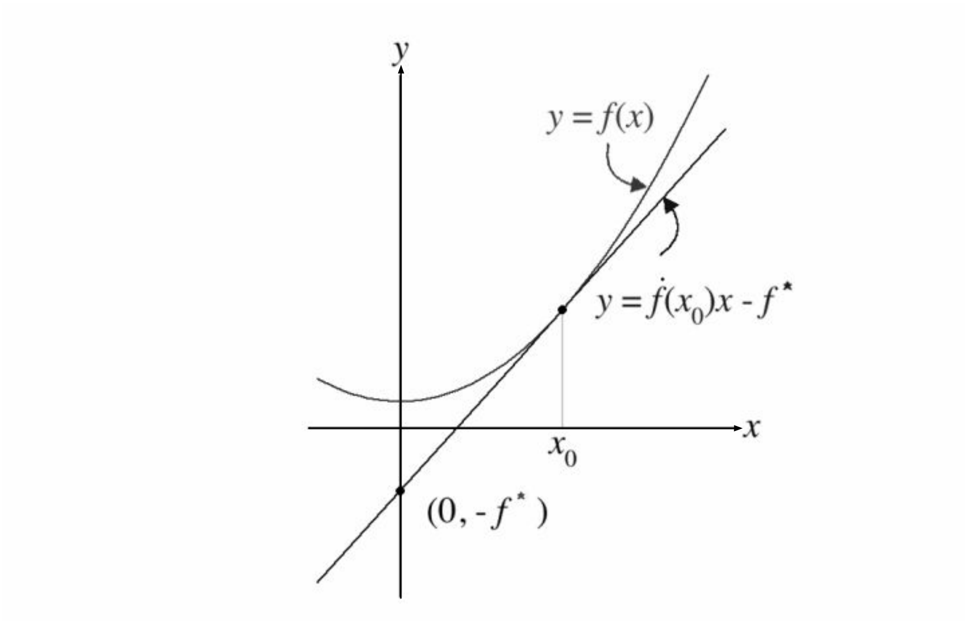


Figure 2.7: Illustration of the Legrendre Fenchel Transform. Figure adapted from Handa et al. [2011].

In the Legrendre Fenchel transform, the points of the function f are transformed into slopes of f^*, and slopes of f and transformed into points of f^*. The Legendre-Fenchel transform is more general than the Legendre transform because it is also applicable to non-convex functions as well as non-differentiable functions. The Legendre-Fenchel transform reduces to Legendre transform for convex functions. Below we discuss some common functions for which the transformation is applied to. The g-weighted Huber norm model is of particular interest as it is used in our formulation in Chapters 3 and 4.

g-Weighted Squared L2 Norm Model: The squared L2 norm applies a quadratic penalty to the error. This function can be additionally multiplied with a scalar weight g for suppressing the penalization magnitude.

We can write the g-weighted squared L2 norm model as follows:

$$f(x) = \frac{g}{2}||x||_2 = \frac{g}{2}\sum_i x_i^2 \tag{2.23}$$

The convex conjugate of the g-weighted L2 norm is as follows:

$$f^*(p) = \frac{1}{2g}||p||_2^2 = \frac{1}{2g}\sum_i p_i^2 \tag{2.24}$$

g-Weighted L1 Norm Model: The L1 norm is a commonly used *robust* norm function. It does not penalize outliers as much as the L2 norm. The g-weighted L1 norm that we will discuss below, is additionally multiplied with a weight g for suppressing the error magnitude.

We can write the g-weighted L1 norm model as follows:

$$f(x) = g||x||_1 = g\sum_i |x_i| \tag{2.25}$$

The convex conjugate of the g-weighted L1 norm is as follows:

$$f^*(p) = \begin{cases} 0 & \text{if } ||p||_2 \leq g \\ \infty & \text{otherwise} \end{cases} \tag{2.26}$$

g-Weighted Huber Norm Model: The Huber norm applies squared L2 norm on small errors and L1 norm on large errors so that outliers will not have much of an influence on the solution. The g-weighted Huber norm that we will discuss below, is additionally multiplied with a weight g for suppressing the error magnitude in instances where penalization should not occur, e.g. large depth gradients at depth discontinuities should *not* be penalized.

We can write the g-weighted Huber norm model as follows:

$$f(x) = g||x||_\epsilon = \begin{cases} \dfrac{g||x||_2^2}{2\epsilon} & \text{if } ||x||_2 \leq \epsilon \\ g||x||_1 - g\dfrac{\epsilon}{2} & \text{otherwise} \end{cases} \tag{2.27}$$

Note that x here can represent a vector or a scalar. In our work in Chapter 3, x represents a vector of two elements.

The convex conjugate of the Huber norm is as follows:

$$f^*(p) = \begin{cases} \dfrac{\epsilon}{2g}||p||_2^2 & \text{if } ||p||_2 \leq g \\ \infty & \text{otherwise} \end{cases} \tag{2.28}$$

Figure 2.8 depicts some examples of the Legrendre-Fenchel transform for L1 norm, squared L2 norm, and Huber norm. Note that unlike the Fenchel Transform the mapping of the Legrendre Fenchel transform is not one-one. Unless f is a convex function, the mapping is infact many-to-one (many primal functions can lead to the same dual function). The double Legrendre Fenchel transform returns the convex envelope f^{**} of the original function f.

2.4 Primal-Dual Optimisation

The first-order primal-dual algorithm is a highly efficient method suitable for solving a class of structured convex optimization problems (Chambolle and Pock [2010]). Most problems in computer vision can be expressed in the form of energy minimisations (Chambolle and Pock [2010]). The variables are defined spatially in an image grid and the regularization terms usually perform some operation on a select set of the variables in this grid, for example computing the spatial gradients of the signal. A general class of the functions representing these problems can be written as:

$$\min_y F(Ky) + G(y) \tag{2.29}$$

where F and G are proper convex functions and $K \in \mathcal{R}^{nxm}$. $F(Ky)$ corresponds to regularization term of the form $||Ky||$ and $G(y)$ corresponds to the data term. F is a convex (continuous) function

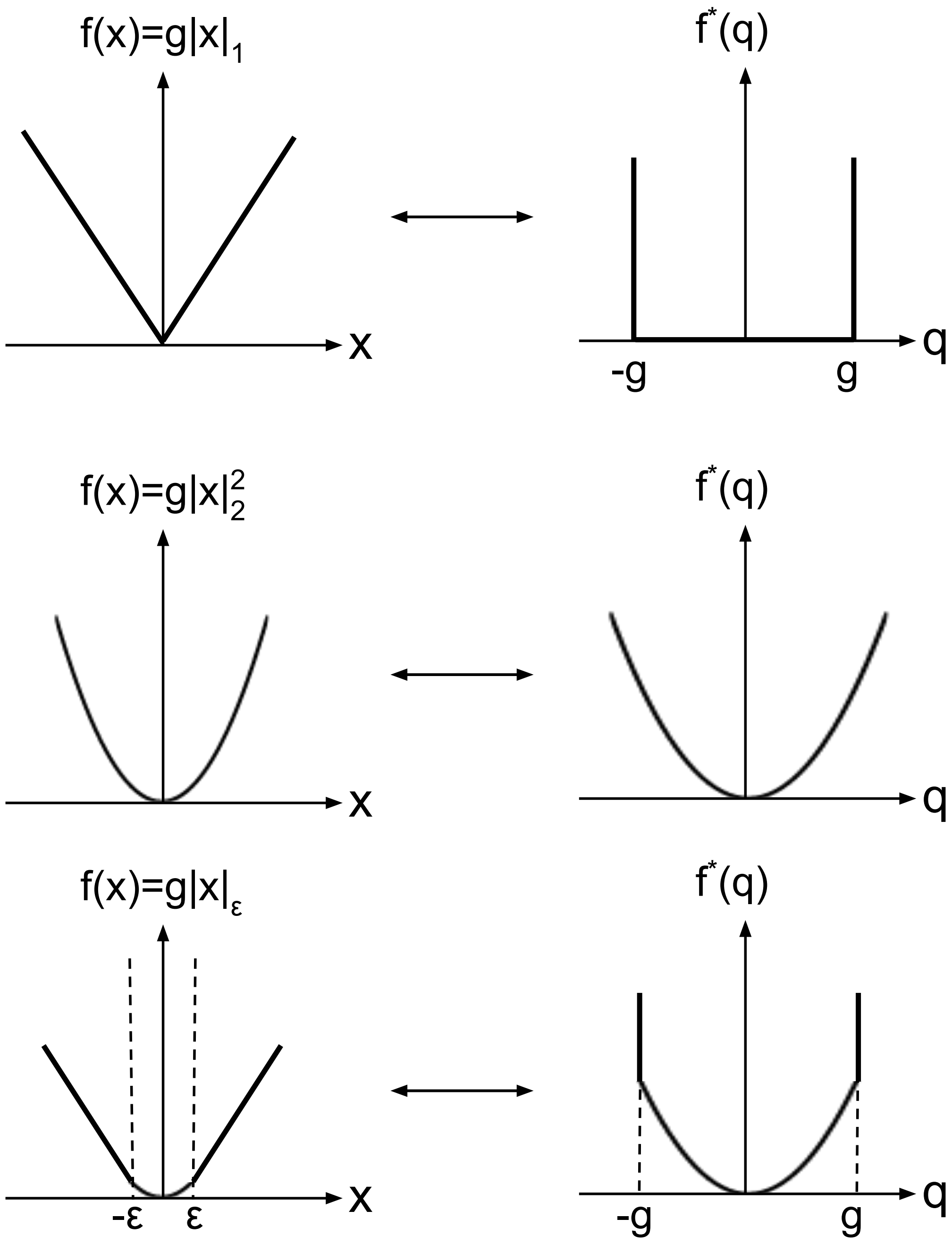


Figure 2.8: Figure depicts (from top to bottom) some examples of the Legrendre-Fenchel Transform (convex conjugate) for L1 norm, squared L2 norm, and Huber norm respectively.

and therefore its Legrendre-Fenchel transformation is:

$$F(Ky) = \max_{q} \langle Ky, q \rangle - F^*(q) \tag{2.30}$$

Objective (2.29) can now be written in the primal-dual form by replacing $F(Ky)$ with its convex conjugate:

$$\min_{y} \max_{q} \langle Ky, q \rangle - F^*(q) + G(y) \tag{2.31}$$

The Primal-Dual Gap: We can derive the primal-dual gap as follows. We know that the dot product is commutative so we can re-write

$$\langle Ky, q \rangle = \langle y, K^T q \rangle \tag{2.32}$$

and in the case the dot product is defined on hermitian space we can write it as

$$\langle Ky, q \rangle = \langle y, K^* q \rangle \tag{2.33}$$

where K^* is the adjoint conjugate of K which is more general. Going back to Equation (2.31), the equation now becomes:

$$\min_{y} \max_{q} \langle y, K^* q \rangle - F^*(q) + G(y) \tag{2.34}$$

By definition,

$$\min_{y} \langle y, K^* q \rangle + G(y) = -G^*(-K^* q) \tag{2.35}$$

because

$$\max_{y} \langle y, K^* q \rangle - G(y) = G^*(K^* q) \tag{2.36}$$

$$\min_{y} -\langle y, K^* q \rangle + G(y) = -G^*(K^* q) \tag{2.37}$$

$$\min_{y} -\langle y, -K^* q \rangle + G(y) = -G^*(-K^* q) \tag{2.38}$$

$$\min_{y} \langle y, K^* q \rangle + G(y) = -G^*(-K^* q) \tag{2.39}$$

Under the weak assumptions in convex analysis, min and max can be switched in Equation (2.39):

$$\langle y, K^* q \rangle + G(y) = \max_{q} -G^*(-K^* q) \tag{2.40}$$

Substituting Equation (2.40) in Equation (2.34), the dual problem then becomes:

$$\max_{q} -G^*(-K^* q) - F^*(q) \tag{2.41}$$

The primal dual gap can be then defined as:

$$\min_{y} F(Ky) + G(y) - \max_{q} -G^*(-K^* q) - F^*(q) \tag{2.42}$$

Algorithm Overview: For the primal-dual algorithm to be applicable, one should be able to compute the proximal mapping of F^* and G, defined as:

$$Prox_{\gamma F}(y) = \arg\min_{u} \frac{1}{2}||y-u||^2 + \gamma F(u) \tag{2.43}$$

Therefore, one can formulate the minimisation steps as

$$\begin{aligned} q^{n+1} &= Prox_{\sigma F^*}(q^n + \sigma K\overline{y}^n) && \text{(dual proximal)} \\ y^{n+1} &= Prox_{\tau G}(y^n - \tau K^* q^{n+1}) && \text{(primal proximal)} \\ \overline{y}^{n+1} &= y^{n+1} + \theta(y^{n+1} - y^n) && \text{(extrapolation)} \end{aligned} \tag{2.44}$$

Note that the proximal mappings reduce to standard pointwise gradient ascent/descent steps. A possible projection onto a unit ball (by division by $max(1, ||q||_2)$ as seen in the example further below) is necessary where F is L1/Huber norm for instance.

It can be shown that if $0 \leq \theta \leq 1$ and $\sigma\tau||K||^2 < 1$, y^n converges to the minimizer of the original energy function. The algorithm's convergence rate depends on different types of the problem. For a completely non-smooth problem: $O(1/N)$ for the duality gap, sum of a smooth and non-smooth: $O(1/N^2)$ and completely smooth problem: $O(1/e^N)$, where N is the desired precision, such that $x - x^* < N$. Note that in our experiments we use the Arrow-Hurwicz method (by setting $\theta = 0$ in Equation (2.44)) following DTAM (Newcombe et al. [2011b]), i.e. without the extrapolation/momentum step. Theoretically by doing so we lose the $O(1/N^2)$ convergence guarantee but in practice this method is shown to be the fastest in Chambolle and Pock [2010].

A Practical Example: We now particularly look at the case when $F(Ky)$ is the Huber norm function as we employ it in our formulation in Chapters 3 and 4. The Huber norm function is *continuous* and its first-derivative is defined at all points, so a simple gradient descent on the function can bring us to the minima. However, since its first-derivative is non-continuous, if we were to use Newton-Raphson method which requires the second order derivative, it wouldn't be possible to do so. Therefore the primal dual optimization method is preferred in this case for faster convergence and efficient parallel implementation (Chambolle and Pock [2010]).

Consider the following objective:

$$\min_{\rho} \sum_{p} \frac{1}{2\theta}||\rho_p - \rho_p^{obs}||_2^2 + g_p||\nabla_p^{\rho}||_{\epsilon} \tag{2.45}$$

where ρ_p^{obs} is our observed value (e.g. inverse depth) for a pixel p in the keyframe image, and ∇_p^{ρ} is a vector of operations where each element is a linear function of ρ_p and one (or more) of its spatial neighbors, e.g. $\nabla_p^{\rho} = \nabla\rho_p$ (the spatial gradients of ρ_p). The first term in the expression corresponds to the observation model and the second term corresponds to the prior model. The parameter θ controls the influence of each of the models on the solution. In Chapter 3 we introduce a novel ∇_p^{ρ}, a vector of operations involving the predicted surface normal at pixel p and the inverse depth values in the local neighborhood of that surface.

Writing $g_p||\nabla_p^{\rho}||_{\epsilon}$ in its dual form based on its convex conjugate, we obtain:

$$g_p||\nabla_p^\rho||_\epsilon = \langle q_p, \nabla_p^\rho\rangle - f^*(q_p) = \langle q_p, \nabla_p^\rho\rangle - \delta_q(\frac{q_p}{g_p}) - \frac{\epsilon}{2g_p}||q_p||_2^2 \tag{2.46}$$

The objective can now be written as:

$$\min_\rho \max_q \sum_p \frac{1}{2\theta}||\rho_p - \rho_p^{obs}||_2^2 + \langle q_p, \nabla_p^\rho\rangle - \delta_q(\frac{q_p}{g_p}) - \frac{\epsilon}{2g_p}||q_p||_2^2 \tag{2.47}$$

Differentiating the above objective with respect to q_p gives:

$$\nabla_p^\rho - \frac{\epsilon}{g_p}q_p \tag{2.48}$$

Differentiating the objective with respect to ρ_p gives:

$$\frac{1}{\theta}(\rho_p - \rho_p^{obs}) + \nabla_p^q \tag{2.49}$$

Note ∇_p^q is a linear operation resulting in a scalar which is computed based on q_p and one or more of its neighbors. This linear operation resulting in a scalar is dependent on the definition of the vector of operations ∇_p^ρ. For example, if $\nabla_p^\rho = \nabla\rho_p$ (the spatial gradients of ρ_p) then $\nabla_p^q = -divq$ (the negative divergence of q_p).

Performing gradient ascent on the dual variable q_p with stepsize σ_q gives:

$$\begin{aligned}\frac{q_p^{n+1} - q_p^n}{\sigma_q} &= (\nabla_p^\rho)^n - \frac{\epsilon}{g_p}q_p^{n+1} \\ \Rightarrow q_p^{n+1} &= g_p \prod(\frac{q_p^n + \sigma_q(\nabla_p^\rho)^n}{g_p + \sigma_q\epsilon})\end{aligned} \tag{2.50}$$

where $\prod(x) = x/max(1, ||x||_2)$ is the projection operation that projects x onto a unit ball.

Performing gradient descent on the primal variable ρ_p with stepsize σ_ρ gives:

$$\begin{aligned}\frac{\rho_p^{n+1} - \rho_p^n}{\sigma_\rho} &= -((\nabla_p^q)^{n+1} + \frac{1}{\theta}(\rho_p^{n+1} - \rho_p^{obs})) \\ \Rightarrow \rho_p^{n+1} &= \frac{\rho_p^n + \sigma_\rho(-(\nabla_p^q)^{n+1} + \frac{1}{\theta}\rho_p^{obs})}{1 + \frac{\sigma_\rho}{\theta}}\end{aligned} \tag{2.51}$$

Benefits of Using the Primal-Dual Algorithm: An immediate benefit of the primal-dual algorithm is that it can be applied when the objective function is non-smooth (but still continuous), i.e. where methods such as the conjugate gradient algorithm cannot be applied. A closer analysis highlights some additional benefits.

Being able to compute the proximal mapping of F is equivalent to being able to compute the proximal mapping of F^* due to the Moreau's identity:

$$y = Prox_{\tau F^*}(y) + \tau Prox_{F/\tau}(y/\tau) \tag{2.52}$$

One of the other main benefits of the primal-dual algorithm is that the proximal mapping of F^* is most often much simpler to compute than that of F, especially in case F is a non-smooth function, like the L1 norm.

Dualizing F also allows for point-wise separable primal and dual variable updates. Consider the example where F computes the spatial gradients of variables arranged in a spatial grid (e.g. values of an inverse depth map). If we didn't dualize and applied the Euler-Lagrange method instead, we would need to compute the second order spatial gradient of the variables (with a kernel representing the Laplacian operator) in order to update each variable. This means that, at each variable update step, the solution of each variable is dependent on the state of its spatial neighbors.

In contrast, when solving the above example with the primal-dual method, updating the dual variable involves performing the gradient operation on the primal variable, and updating the primal variable involves performing the negative divergence operation on the dual variable. Hence the primal and dual problem at each update step is point-wise separable (i.e. the primal/dual solution is independent of the state of the spatial neighbours of the primal/dual variables respectively). This allows for in-place operations and easy GPU parallelization.

The primal-dual algorithm also bears optimal convergence rate on multiple sub-classes compared to methods such as ADMM (Alternating Direction Method of Multipliers), PGM (Proximal Gradient Method), etc. For a fully convex and smooth objective, primal dual optimization brings us closer to the solution very quickly due to the strong duality (Chambolle and Pock [2010]). A detailed comparison and derivation of convergence rates for the primal-dual algorithm and each baseline method can be found in Chambolle and Pock [2010].

2.5 Summary

Present monocular visual SLAM solutions are unable to satisfy one or more of the following desired attributes: (1) accurate and dense 3D mapping and precise camera localisation, (2) large-scale operation, (3) real-time low latency execution, (4) long-term consistent performance, (5) static scene mapping in dynamic environments (while modelling dynamic objects if necessary), (6) efficient and compact 3D model representation, and (7) effective utilization of *learned* priors and stored compact prior scene knowledge for rapid and high-quality reconstruction. These desired attributes are inter-related, and in this thesis we mainly focus on attribute (7). Some works (Dame et al. [2013]) have looked into object level priors for improving reconstruction density and accuracy in monocular SLAM however for a limited set of classes, while others (Flint et al. [2011]) have looked at improving dense reconstructions with Manhattan world priors. More general scene level priors such as inverse depth smoothness (Newcombe [2012]) are primarily handcrafted and less than ideal.

Monocular visual SLAM approaches primarily rely on multi-view geometry to solve for depth and camera pose with great success. However, performance is limited in areas of uniform texture, when there are local and global illumination changes, appearance distortion, etc. This is mainly because these approaches (especially direct SLAM methods) rely on low-level RGB/hand-crafted features for matching, an issue we address in our work by *learning* good features to match. Recent works have proposed methods to use CNNs for deep metric feature learning (Choy et al. [2016]), but these features are not trained and extensively tested in the context of dense monocular SLAM, and the network architectures are not specifically designed for real-time feature extraction.

Recently, depth and surface normal predictions from Convolutional Neural Networks (CNNs) have become increasingly accurate, and CNNs have also been used for predicting depth, optical flow, and camera pose from two images (Kendall et al. [2017]; Ummenhofer et al. [2017]). Unsupervised methods for geometry estimation have also become a popular research area (Garg et al. [2016]; Zhan et al. [2018]). Few works (Bloesch et al. [2018]; Tateno et al. [2017]), similar to ours, have also taken steps at combining learned geometry estimates with exisiting dense monocular SLAM methods.

Current sparse SLAM methods store the reconstructed model as a raw point cloud. Dense methods typically store the reconstructed model using a TSDF-based voxel array (Newcombe et al. [2011a]) or using surfels (Whelan et al. [2011]), while some methods replace objects in the map with CAD models (Salas-Moreno et al. [2013]). Finding efficient geometric and semantic embeddings for a reconstructed dense map is an open research area and complements our "just-in-time" reconstruction strategy advocated in this thesis. Recent work (Bloesch et al. [2018]) suggest this compression can be achieved by encoding keyframe depth maps into low dimensional embeddings through the use of a convolutional auto-encoder conditioned on keyframe images.

Bibliography

P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

H. Alismail, B. Browning, and S. Lucey. Direct visual odometry using bit-planes. *arXiv preprint arXiv:1604.00990*, 2016a.

H. Alismail, B. Browning, and S. Lucey. Photometric bundle adjustment for vision-based slam. In *Asian Conference on Computer Vision*, pages 324–341. Springer, 2016b.

A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan. Pixelnet: Towards a general pixel-level architecture. *arXiv preprint arXiv:1609.06694*, 2016a.

A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5965–5974, 2016b.

S. Y. Bao and S. Savarese. Semantic Structure from Motion: A Novel Framework for Joint Object Recognition and 3D Reconstruction. In *Proceedings of the 15th International Conference on Theoretical Foundations of Computer Vision: Outdoor and Large-scale Real-world Scene Analysis*, pages 376–397, Berlin, Heidelberg, 2012.

S. Y. Bao, M. Chandraker, Y. Lin, and S. Savarese. Dense Object Reconstruction with Semantic Priors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1264–1271, jun 2013.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam-learning a compact, optimisable representation for dense visual slam. *arXiv preprint arXiv:1804.00874*, 2018.

L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

M. J. Brooks and B. K. Horn. Shape and source from shading. 1985.

C. Cadena, A. R. Dick, and I. D. Reid. Multi-modal auto-encoders as joint estimators for robotics scene understanding.

M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

A. Chambolle and T. Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2010.

A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Structure from motion causally integrated over time. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):523–535, 2002.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems 30*. 2016.

A. Concha and J. Civera. Using superpixels in monocular SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 365–372, may 2014.

A. Concha and J. Civera. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5686–5693, sep 2015.

A. Concha, M. W. Hussain, L. Montano, and J. Civera. Manhattan and Piecewise-Planar Constraints for Dense Monocular Mapping. In *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, 2014.

A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 434–441. IEEE, 1999.

M. Cummins and P. Newman. Invited Applications Paper FAB-MAP: Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model. In *27th Intl Conf. on Machine Learning (ICML2010)*, 2010.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid. Dense Reconstruction Using 3D Object Shape Priors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1288–1295, jun 2013.

A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

T. Dharmasiri, A. Spek, and T. Drummond. Joint prediction of depths, normals and surface curvature from rgb images using cnns. *arXiv preprint arXiv:1706.07593*, 2017.

T. Dharmasiri, A. Spek, and T. Drummond. Eng: End-to-end neural geometry for robust depth and pose estimation using cnns. *arXiv preprint arXiv:1807.05705*, 2018.

A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

J.-D. Durou, Y. Quéau, and J.-F. Aujol. Normal Integration – Part I: A Survey. jun 2016. URL https://hal.archives-ouvertes.fr/hal-01334349.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

E. Eade and T. Drummond. Scalable monocular slam. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 469–476. IEEE Computer Society, 2006.

D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

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*, pages 2366–2374, 2014.

J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

A. Flint, D. Murray, and I. Reid. Manhattan scene understanding using monocular, stereo, and 3D features. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2228–2235, nov 2011.

C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3392–3399. IEEE, 2013.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

C. Godard, O. Mac Aodha, and G. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611. IEEE, 2017.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006.

A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Applications of legendre-fenchel transformation to computer vision problems, 2011.

C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys. Joint 3D Scene Reconstruction and Class Segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 97–104, jun 2013.

C. Hane, N. Savinov, and M. Pollefeys. Class Specific 3D Object Shape Priors Using Surface Normals. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014.

C. Hane, L. Ladicky, and M. Pollefeys. Direction matters: Depth estimation with a surface normal classifier. In *CVPR*, pages 381–389, 2015.

C. Harris and M. Stephens. A combined corner and edge detector. Citeseer, 1988.

C. G. Harris and J. Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM transactions on graphics (TOG)*, volume 24, pages 577–584. ACM, 2005.

D. Hoiem, A. A. Efros, and M. Hebert. Recovering Surface Layout from an Image. *Int. J. Comput. Vision*, 75(1):151–172, Oct 2007.

J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. *CoRR*, abs/1803.08673, 2018. URL http://arxiv.org/abs/1803.08673.

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.2, 1962.

A. Kanazawa, D. W. Jacobs, and M. Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3253–3261, 2016.

A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6555–6564. IEEE, 2017.

A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590, 2017.

A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 66–75, 2017.

H. Kim. *Real-time visual SLAM with an event camera.* PhD thesis, Imperial College London, UK, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007a.

G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007b.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques.* 2009.

K. Kraus. *Photogrammetry: Geometry from Images and Laser Scans.* De Gruyter textbook. Walter De Gruyter, 2007. ISBN 9781680152616.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

A. Kundu, Y. Li, F. Dellaert, F. Li, and J. Rehg. Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In *Computer Vision – ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pages 703–718. 2014.

O. Kähler, V. Prisacariu, J. Valentin, and D. Murray. Hierarchical voxel block hashing for efficient integration of depth images. *IEEE Robotics and Automation Letters*, 1(1):192–197, Jan 2016. ISSN 2377-3766.

L. Ladický, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. Torr. Joint Optimization for Object Class Segmentation and Dense Stereo Reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012.

L. Ladický, B. Zeisl, and M. Pollefeys. Discriminatively Trained Dense Surface Normal Estimation. In *ECCV*, volume 8693, pages 468–484. 2014.

I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. In *Dense Image Correspondences for Computer Vision*, pages 15–49. Springer, 2016a.

F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38 (10):2024–2039, 2016b.

G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. *arXiv preprint arXiv:1804.07723*, 2018.

J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2014.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

F. Ma and S. Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *arXiv preprint arXiv:1709.07492*, 2017.

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

M. J. Milford and G. F. Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1643–1649, may 2012.

M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. 2002.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

R. Newcombe. *Dense visual SLAM*. PhD thesis, Imperial College London, UK, 2012.

R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *IEEE ISMAR*. IEEE, 2011a.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011b.

R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):169, 2013.

D. Nister. An efficient solution to the five-point relative pose problem. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–195. IEEE, 2003.

H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4797–4805. Curran Associates Inc., 2016.

D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2609–2616. IEEE, 2014.

V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. MonoFusion: Real-time 3D Reconstruction of Small Scenes with a Single Web Camera. In *ISMAR*, 2013.

V. Prisacariu, O. Kahler, M. Cheng, C. Ren, J. Valentin, P. Torr, I. Reid, and D. Murray. A Framework for the Volumetric Integration of Depth Images. *ArXiv e-prints*, 2014.

R. T. Rockafellar. *Convex Analysis*. Princeton landmarks in mathematics and physics. Princeton University Press, 1997.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5506–5514, 2016.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013.

A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009.

T. Schmidt, R. Newcombe, and D. Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017.

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

H. Strasdat, J. Montiel, and A. J. Davison. Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. IEEE, 2010a.

H. Strasdat, J. Montiel, and A. J. Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2, 2010b.

H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige. Double window optimisation for constant time visual slam. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359. IEEE, 2011.

D. Strong and T. Chan. Edge-preserving and scale-dependent properties of total variation regularization. *Inverse problems*, 19(6):S165, 2003.

J. Stühmer, S. Gumhold, and D. Cremers. Real-time Dense Geometry from a Handheld Camera. In *Proceedings of the 32Nd DAGM Conference on Pattern Recognition*, pages 11–20, Berlin, Heidelberg, 2010. Springer-Verlag.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

S. Suwajanakorn, C. Hernandez, and S. M. Seitz. Depth from focus with your mobile phone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3497–3506, 2015.

C. Tang and P. Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.

K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6565–6574. IEEE, 2017.

S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Y. Ng. Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In *Algorithmic Foundations of Robotics V*, pages 363–380. Springer, 2004.

E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2010.

B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment—A Modern Synthesis. In *International Workshop on Vision Algorithms*, pages 298–372. Springer, 1999.

B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017.

A. Valada, N. Radwan, and W. Burgard. Deep auxiliary learning for visual localization and odometry. *arXiv preprint arXiv:1803.03642*, 2018.

S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille. SURGE: Surface Regularized Geometry Estimation from a Single Image. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 172–180. 2016.

S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2043–2050. IEEE, 2017.

X. Wang, D. F. Fouhey, and A. Gupta. Designing Deep Networks for Surface Normal Estimation. In *CVPR*, 2015.

T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: dense slam without a pose graph. Robotics: Science and Systems, 2011.

J. Xie, R. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.

Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.

K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: learned invariant feature transform. *CoRR*, abs/1603.09114, 2016. URL http://arxiv.org/abs/1603.09114.

S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015.

J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1592–1599, 2015.

M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. 2018.

R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analaysis and Machine Intelligence*, 21(8):690–706, 1999.

Y. Zhang and T. Funkhouser. Deep depth completion of a single rgb-d image. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

H. Zhou, B. Ummenhofer, and T. Brox. Deeptam: Deep tracking and mapping. *arXiv preprint arXiv:1808.01900*, 2018.

T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6612–6619. IEEE, 2017.

Chapter 3

Dense Monocular Reconstruction using Surface Normals

In this chapter we introduce our learned surface normal prior for dense keyframe reconstruction, that is fused with information from multi-view photo-consistency, alongside a framework for real-time incremental monocular dense reconstruction. We first provide a brief introduction to the relevant problem, and revisit some related background work. We then move onto introducing our methodology, followed by an extensive evaluation primarily comparing our method against the performance of the weaker smoothness prior of DTAM. At the end of the chapter we discuss some potential future directions.

3.1 Introduction

The ability to carry out dense and accurate 3D mapping of the environment is desirable in applications such as autonomous navigation, robotic manipulation, augmented reality, etc. Doing so merely using input from a moving monocular/stereo camera is attractive as cameras are ubiquitous, compact, power efficient, and not limited by range. Visual SLAM (Simultaneous Localisation and Mapping) systems have evolved to the point where they are capable of many feats such as accurate and real-time camera tracking and 3D mapping (Engel et al. [2014]; Klein and Murray [2007]; Mur-Artal et al. [2015]), with some capable of fully dense live 3D reconstruction (Newcombe et al. [2011]; Stühmer et al. [2010]).

Less attention has been paid to the application of high-level scene understanding to aid the mapping process. Some works have introduced constraints such as Manhattan world assumptions, or piecewise planar priors (Concha and Civera [2014, 2015]; Concha et al. [2014]; Flint et al. [2011]). Other stronger priors have also been leveraged, such as known objects (Bao and Savarese [2012]; Bao et al. [2013]; Dame et al. [2013]; Hane et al. [2013]; Kundu et al. [2014]; Ladický et al. [2012]), while other works have made use of smoothness assumptions to "fill in" regions where there is insufficient photometric variation (Herrera C. et al. [2013]; Newcombe et al. [2011]; Pizzoli et al. [2014]; Pradeep et al. [2013]). For example, DTAM (Newcombe et al. [2011]) requires hand crafted priors on depth (piecewise constant inverse depth) to fill in the regions of low texture. High-level scene context (e.g. offices having a desk, on top of which a monitor, keyboard, etc. often lie in a specific configuration) is usually ignored in pure-geometry based SLAM systems.

In our work we recognise that recent advances in deep learning techniques mean that priors about

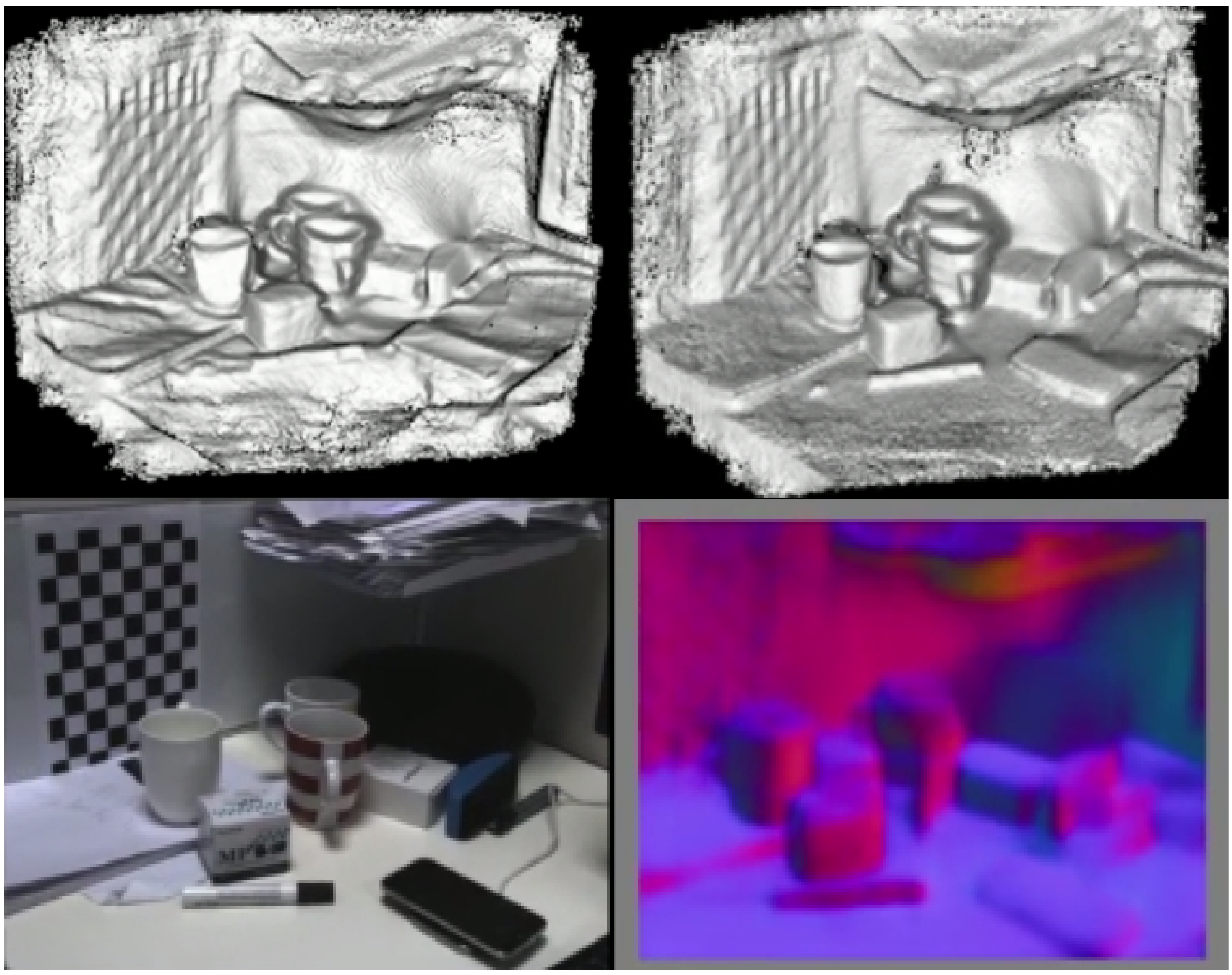


Figure 3.1: Reconstruction using smoothness regularizer (top left) and normal prior (top right), an RGB image in the sequence (bottom left), and corresponding predicted normals (bottom right). Note the more accurate high-fidelity reconstruction obtained using the normal prior. A live comparison video is available at https://youtu.be/atq1EhX-75k.

surface orientation can be captured within a multi-layer Convolutional Neural Network (CNN) which regresses image patches to local surface normal values (Eigen and Fergus [2015]; Liu et al. [2015a]). The predictions are likely to be backed by the capacity of neural networks to grasp high-level concepts such as object type and scene layout, and relative orientation to one another, in addition to low-level cues such as shading (Zhang et al. [1999]). The improvements they can bring to traditional reconstruction techniques are also demonstrated in recent work like Hane et al. [2015]. Motivated by this, we here present the first framework for real-time monocular dense mapping that efficiently combines both the benefits of deep CNNs and well-established geometry-based methods.

In particular, we use normal predictions from a learned neural network (Eigen and Fergus [2015]) as a strong prior and aim to estimate a map which (i) minimizes photometric cost and (ii) is consistent with the single-view normal predictions. To that end we incorporate a depth/normal consistency term in our energy minimization framework which acts as a regularizer to fill in the gaps in the map with very little texture. We closely follow DTAM (Newcombe et al. [2011])'s energy minimization method for mapping where our proposed regularizer replaces their inverse-depth smoothness prior.

We extensively evaluate our proposed method quantitatively against the pure geometry-based reconstructions of DTAM, and pure learning-based depth predictions of Eigen and Fergus [2015], on a diverse range of indoor sequences in a large dataset like raw NYU-D V2 (Silberman et al. [2012]). Our benchmarking highlights limitations of both pure learning-based and pure-geometry-based systems, which are addressed by the proposed method.

3.2 Background and Related Work

An early separate line of work aimed to reconstruct purely based on surface normal information (Durou et al. [2016]). For instance in Kovesi [2005], input surface normals are used to reconstruct surfaces by correlating them with a set of basis functions, referred to as shapelets. The surface gradients of the shapelets are correlated with the gradients of the surface and the correlations are summed to form the reconstruction. A similar approach was taken in Framkot and Chellappa [1988] where a possibly non-integrable estimate of surface slopes is represented by a finite set of basis functions in Fourier Domain, and integrability is enforced by calculating the orthogonal projection onto a vector subspace spanning the set of integrable slopes. The integrability projection constraint was applied by extending the iterative shape-from-shading algorithm of Brooks and Horn [1985].

More closely aligned with our work, in Kolev et al. [2010], a formulation to integrate normals with photometric consistency for reconstruction was proposed. In their experiments however, the normal field F was computed based on local-planar patch fitting that maximized photo-consistency, while on the other hand we use *learned* normals. They propose to minimize the following energy with respect to the estimated surface S:

$$E(S) = \int_S \sqrt{N_S(s)^T D(s) N_S(s)}\, ds \tag{3.1}$$

where

$$D(s) = \rho^2(\tau F F^T + \frac{3-\tau}{2}(I - FF^T)) \tag{3.2}$$

where τ is a weighting parameter that controls the tolerance of the normal field F. The classical weighted minimal surface model appears as a special case when $\tau = 1$ and $D = \rho^2 I$, and penalization is zero along the normal field F when $\tau = 0$. Our formulation shares the same idea as theirs, however the difference lies in the fact that we minimize the dot product between a given set of surface normals and corresponding *difference vectors in 3D* parameterized by the inverse depth map of a keyframe (as we explain later on), while they minimize the negative dot product between a given set of surface normals, F and the corresponding set of surface normals of the estimated 3D surface, $N_S(s)$. Their formulation is suited for volumetric reconstruction in general, while ours is tailored for keyframe reconstruction with inverse depth parameterization.

Another difference lies in the fact that their photoconsistency terms ρ are multiplied into the normal prior terms while in our case the photoconsistency terms and normal prior terms are linked via summation of energies. While it remains to be shown which strategy works best, linking via a *separable sum* of energies offers some benefits: (1) use of a scalar constant to balance the contribution of the photo-consistency terms and the normal-based prior terms, (2) the ability to easily separate the overall objective into a convex and non-convex parts which is helpful for optimization purposes, and (3) the practical benefit of the being able to independently train for the uncertainties of the normal-based prior model (as discussed in this chapter) and the photo(or feature)-consistency model (as discussed in Chapter 4).

In Hane et al. [2014], in a direction similar to ours, 3D shape priors were formulated based on *learned* surface normals for a set of object classes. Wulff shapes parameterized by surface normals of the object surface (for each occupied voxel in the volume) were trained for each class and they determined the amount of directional regularization of the reconstructed surface. Their method however was applied for volumetric reconstruction of objects rather than scenes, and is not specifically tailored for keyframe reconstruction. More recently, in Hane et al. [2015], a similar method as in Hane et al. [2014] was used for refining single-image depth predictions, or stereo-based depth estimates using surface normal predictions from a neural network. Pixel-wise classification scores of a discrete set of *learned* surface normals were used to pre-compute a Wulff shape for each pixel in the keyframe which in turn determined the amount of directional smoothing of the reconstructed surface. Note that constructing a Wulff shape based on the orientation of each point in the reconstructed surface introduces additional computational and storage overhead. In comparison our regularizer is a direct function of both the normals and the inverse depth solution. Moreover, the experiments in Hane et al. [2015] were limited to improving stereo reconstructions and single-view depth predictions, and the formulation utilized discretized normal predictions.

3.3 Method

Our proposed framework incrementally generates a fully dense reconstruction of a scene from a video sequence given pixel-wise surface normal maps of keyframes and photometric evidence from a series of overlapping images corresponding to those keyframes. The following sub-sections elaborate on the key components.

3.3.1 Notations

The following notations and conventions will be used in this section. K is the camera intrinsic matrix. $\mathbf{I}_r \in \mathbb{R}^3$ is a $M \times N$ keyframe (reference) image and $\mathbf{I}_n \in \mathbb{R}^3$ is a $M \times N$ image in the set of images overlapping $\mathbf{I}_r$. We assume images are undistorted. $\mathbf{u}_p = (u, v)^T$ is a pixel location in $\mathbf{I}_r$, where $p = 1, ..., MN \in \mathcal{P}$ is a pixel location-based index. $\dot{\mathbf{u}}_p = (u, v, 1)^T$ is $\mathbf{u}_p$ in homogeneous form, and $\tilde{\mathbf{x}}_\mathbf{p} := K^{-1}\dot{\mathbf{u}}_p$. For a pixel $\mathbf{u}_p$, d_p and ρ_p are the corresponding depth and inverse-depth respectively, and $\mathbf{d}$ and $\boldsymbol{\rho}$ are $MN \times 1$ vectors of stacked d_p and ρ_p values respectively. $T_{nr} \in \mathbf{SE}(3)$ is a matrix describing the transformation of a point from camera coordinates of $\mathbf{I}_r$ to that of $\mathbf{I}_n$. $\pi(.)$ and $\pi^{-1}(.,.)$ are the projection and back-projection operations, such that $\pi(K^{-1}\dot{\mathbf{u}}_p/\rho_p) = \mathbf{u}_p$ and $\pi^{-1}(\mathbf{u}_p, \rho_p) = K^{-1}\dot{\mathbf{u}}_p/\rho_p$. The predicted surface normal vectors $\hat{\mathbf{n}}_p \in \mathbb{R}^3$ are normalized in Euclidean space and are in camera coordinates of $\mathbf{I}_r$.

3.3.2 Energy Formulation for Keyframe Reconstruction

We formulate depth estimation of a keyframe as an energy minimization problem. Closely following Newcombe et al. [2011] our energy function consists of a dataterm and a regularization term as follows

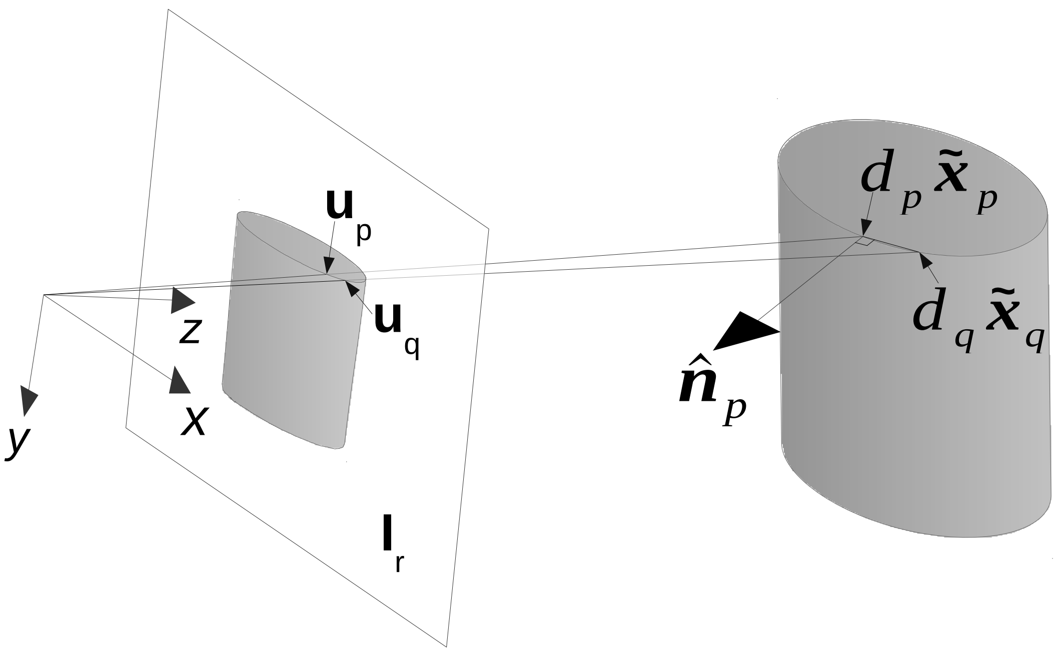


Figure 3.2: A neighbouring 3D point pair and corresponding surface normal.

and will be minimized with respect to $\boldsymbol{\rho}$ (the keyframe inverse depth map):

$$E(\boldsymbol{\rho}) = \sum_{p \in \mathcal{P}} \frac{1}{\lambda} E_\phi(\rho_p) + E_{\hat{n}}(\rho_p), \tag{3.3}$$

where λ controls the regularization strength.

E_ϕ is the dataterm that computes the photometric error for a keyframe $\mathbf{I}_r$ accumulated over N overlapping frames:

$$E_\phi(\rho_p) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{I}_r^{'}(\mathbf{u}_p) - \mathbf{I}_n^{'}(\pi(T_{nr}\pi^{-1}(\mathbf{u}_p, \rho_p))) \right\|_1 \tag{3.4}$$

For added robustness in the photometric matching, we concatenate the RGB channels of $\mathbf{I}_r$ and $\mathbf{I}_n$ with an additional image gradient-based channel, computed using Equation (3.9), forming $\mathbf{I}_r^{'}$ and $\mathbf{I}_n^{'}$ respectively. Using Equation (3.4) a cost volume similar to Newcombe et al. [2011] can be created that stores average photometric error for a discrete set of inverse depth labels $l \in \mathcal{L}$, for each pixel $\mathbf{u}_p$. The proceeding sections contain more information pertaining to cost volume creation.

Our proposed regularization term is based on the relationship between a pair of 3D points and its corresponding normal (Figure 3.2) in the camera coordinates of $\mathbf{I}_r$:

$$\langle \hat{\mathbf{n}}_p, d_q\tilde{\mathbf{x}}_q - d_p\tilde{\mathbf{x}}_p \rangle = 0, \tag{3.5}$$

where $\langle \, , \, \rangle$ is the dot product operator and $q \in \mathcal{N}(p)$ corresponds to a pixel location in the neighbourhood of that of p. Equation (3.5) can be simplified as:

$$\begin{gathered} d_q\langle \hat{\mathbf{n}}_p, \tilde{\mathbf{x}}_q \rangle - d_p\langle \hat{\mathbf{n}}_p, \tilde{\mathbf{x}}_p \rangle = 0 \\ d_q c_{pq} - d_p c_{pp} = 0 \end{gathered} \tag{3.6}$$

where c_{pq} and c_{pp} are constants equal to $\hat{\mathbf{n}}_p \cdot \tilde{\mathbf{x}}_q$ and $\hat{\mathbf{n}}_p \cdot \tilde{\mathbf{x}}_p$ respectively.

We now switch from depth to inverse depth parameterization by dividing Equation (3.6) by $d_p d_q$. The motivation behind this switch is explained at the end of this section.

$$\begin{aligned} \frac{1}{d_p} c_{pq} - \frac{1}{d_q} c_{pp} &= 0 \\ \rho_p c_{pq} - \rho_q c_{pp} &= 0 \end{aligned} \tag{3.7}$$

Thus we can minimize the following energy, penalizing inconsistent inverse depth values:

$$E_{\hat{n}}(\rho_p) = g_p \left\| \boldsymbol{\nabla}_p^{\boldsymbol{\rho}} \right\|_\epsilon \tag{3.8}$$

where $\boldsymbol{\nabla}_p^{\boldsymbol{\rho}}$ denotes a vector of operations as follows:

$$\boldsymbol{\nabla}_p^{\boldsymbol{\rho}} = \begin{bmatrix} \rho_p c_{pi} - \rho_i c_{pp} \\ \rho_p c_{pj} - \rho_j c_{pp} \end{bmatrix}.$$

The indices i and j correspond to pixel locations neighbouring that of p in the *positive* x and y directions in the image plane. Note that we have restricted the neighbourhood pairwise connectivity to the latter for computational efficiency.

The image-edge based weight $g_p = g(\mathbf{I}_r, \mathbf{u}_p)$ where

$$g(\mathbf{I}, \mathbf{u}) = e^{-\alpha \|\nabla \mathbf{I}(\mathbf{u})\|_2^\beta} \tag{3.9}$$

reduces regularization at image edges, under the assumption that these regions align with depth discontinuities. α and β are tunable parameters.

The Huber norm, defined as

$$\|\mathbf{x}\|_\epsilon = \begin{cases} \dfrac{\|\mathbf{x}\|_2^2}{2\epsilon} & \text{if } \|\mathbf{x}\|_2 \le \epsilon \\ \|\mathbf{x}\|_1 - \dfrac{\epsilon}{2} & \text{otherwise} \end{cases}, \tag{3.10}$$

also minimises penalties at surface discontinuities, and makes the overall energy more robust to errors in normal predictions.

In the special case when $c_{pq} = c_{pp} = -1$ which occurs when $\hat{\mathbf{n}} = (0, 0, -1)^T$, i.e. the normal is pointed directly at the camera, Equation (3.8) reduces to the smoothness prior used in Newcombe et al. [2011]. Hence ours is a more general form that can enforce inverse depth relationships for arbitrary surface orientations visible to the camera. Adding more flexibility, we introduce a tunable parameter γ which balances smoothness/normals regularization:

$$\begin{aligned} c_{pq} &\leftarrow (1-\gamma) c_{pq} - \gamma \\ c_{pp} &\leftarrow (1-\gamma) c_{pp} - \gamma \end{aligned} \tag{3.11}$$

where γ is a value between 0 and 1. The parameter γ could also be a function of the normal prediction uncertainty estimated using a technique such as in Gal and Ghahramani [2015] or our method proposed in Section 3.5.

Why does inverse depth parameterization help? Note that inverse depth parameterization is generally preferred for *keyframe reconstruction*. Firstly, it provides more ease in modeling the observations. Regular sampling in inverse depth space roughly corresponds to regular sampling along the epipolar lines in the live images, especially for those images with low-parallax. This makes precision in inverse depth, throughout its solution space, an approximate linear function of the image sensor resolution (implying that precision is less for larger depths). The probability distribution of the observation model will therefore be more symmetrical and homoscedastic (i.e. have a more uniform variance throughout the solution space) in inverse depth parameterization, which allows for easier modeling. In depth parameterization, we are faced with the added task of modeling the inherent heteroscedasticity of the observations. Furthermore, regular sampling in inverse depth is an efficient sampling strategy for generating the cost volume.

Secondly, inverse depth parameterization provides more ease in modeling our normal priors. Note that our normal prior terms are being computed based on variables that are in pixel space. Due to effects of perspective projection, neighboring pixels are projected further apart in 3D when they correspond to landmarks that are further away, making the local planarity assumption weaker for far away neighboring points as they could correspond to different surfaces. Unless the neural network is able to capture the variability of the learned normals for far regions, the predicted normals for such regions are likely to be unreliable. Furthermore since normals in our experiments are learned based on normals from Kinect depth maps in place of groundtruth, this adds to the unreliability of the learned surface normals for far away points (as Kinect depth maps are also unreliable for those points with large depth values). The prior terms involving such normals with higher error will correspondingly have high variance. Therefore in order to make the probability distribution of our normal prior model more homoscedastic, the prior terms must be weighted such that the directional error of a neighboring point pair is penalized less if those two points are further away from the camera.

It turns out however that in the original depth-normal inconsistency expression in Equation (3.6), which is parameterized in depth, the opposite effect occurs. This is because the magnitude of the difference vectors is always larger for point pairs that are further away from the camera (as 3D points are restricted to lie along their respective viewing ray). Hence, for the same directional error in a pair of points, the dot product between their difference vectors and corresponding surface normal will always be larger if these points are further away. What we desire ideally is the reverse effect. Note that the surface normal has a magnitude of 1 (as it is normalized). We don't normalize the magnitude of the difference vectors as it introduces complications when optimizing the resulting energy, and also since normalization would remove any bias in penalization, which is also not what we are after.

We can achieve our desired reverse effect by simply dividing Equation (3.6) by $d_p d_q$. Now we get an expression that is non-linear with respect to depth, however by re-parameterization in inverse depth, the inconsistency term becomes a linear expression with respect to the solution. The Huber norm of the expression results in a *continuous* and *convex* energy function for the normal prior terms. In addition to increasing the homoscedasticity of the resulting normal prior distribution, the continuous and convex energy allows for efficient inference and use of standard optimization techniques, like the primal-dual algorithm that we use to solve our objective (Section 3.3.3), or the conjugate gradient method when the Huber norm is replaced with a squared L2 norm (weighted by learned pixelwise confidence) as suggested at the end of Chapter 5. Note that when the number of pixels in the image is small, and the squared L2 norm is used, the energy could also be inferred in closed-form.

If we were to parameterize our energy in depth, we can, for instance, still assume a multi-variate Gaussian/ Laplace distribution and learn a data-dependent confidence weight to both the observation and prior model such that distant points are modeled with high variance. Note that even in our inverse depth formulation, heterescedasticity in the prior distributions still exist (to a comparatively lesser extent), due to errors in normal predictions regardless of distance from the camera, and other non-idealities of the model, e.g. lack of being able to model depth discontinuities, penalization magnitude being slightly higher for points that are closer to the image centre for the same directional error as we don't normalize the difference vector, etc. This heterescedasticity in the model can be compensated for by a pairwise confidence weight. In Section 3.5.2 we propose a way to learn image-dependent pairwise confidence weights for our normal priors by assuming our model to fit a multi-variate Gaussian distribution, and these weights can replace the image edge-weight g_p which was only used as a heuristic to represent confidence.

Inverse depth parameterization also offers other benefits in Extended Kalman filter (EKF)-based approaches for monocular SLAM (Civera et al. [2008]). For example, under inverse depth parameterization landmarks can be initialized with a reasonable variance and used immediately for tracking. It also makes the measurement model more linear with respect to landmark location state resulting in better convergence of the state vector and better feature location prediction in a live image (Civera et al. [2008]). Note that inverse depth parameterization is not used during bundle adjustment as the map points are given freedom to optimize in 3 dimensions (XYZ) without being restricted to particular viewing rays. This is not a big issue as the landmarks are likely to have been observed by images with large baseline at the time of bundle adjustment, which makes the imbalance in precision of far and near points not as severe as during keyframe reconstruction.

3.3.3 Optimisation of Keyframe Inverse Depths

Our objective can be written as follows:

$$\min_{\boldsymbol{\rho}} E(\boldsymbol{\rho}) = \sum_{p \in \mathcal{P}} \frac{1}{\lambda} E_{\phi}(\rho_p) + g_p \left\| \boldsymbol{\nabla}_p^{\boldsymbol{\rho}} \right\|_{\epsilon} \tag{3.12}$$

Based on the Legendre-Fenchel transform, the convex problem $\min_{\boldsymbol{\rho}} \sum_{p \in \mathcal{P}} g_p \left\| \boldsymbol{\nabla}_p^{\boldsymbol{\rho}} \right\|_{\epsilon}$ is equivalent to (Rockafellar [1997]):

$$\min_{\boldsymbol{\rho}} \max_{\mathbf{q}} \sum_{p \in \mathcal{P}} \left\{ \langle \boldsymbol{\nabla}_p^{\boldsymbol{\rho}}, \mathbf{q}_p \rangle - \delta_q(\frac{\mathbf{q}_p}{g_p}) - \frac{\epsilon}{2} \frac{\|\mathbf{q}_p\|_2^2}{g_p} \right\}, \tag{3.13}$$

where $\mathbf{q}_p = [q_{px}, q_{py}]^T$ is the dual variable and $\mathbf{q} = [\mathbf{q}_1, ..., \mathbf{q}_{MN}]^T$. $\delta_q(\mathbf{q}_p/g_p) = 0$ if $\|\mathbf{q}_p/g_p\|_2 \leq 1$ and ∞ otherwise.

Following Newcombe et al. [2011] and Steinbrücker et al. [2009], we introduce a linking (or coupling) term $\frac{1}{2\theta} \|\rho_p - a_p\|_2^2$ into (3.12) and replace $E_\phi(\rho_p)$ with $E_\phi(a_p)$, where a_p is an auxiliary variable. We define $\mathbf{a}$ to be a vector such that $\mathbf{a} = [a_1, ..., a_{MN}]^T$.

The objective (3.12) can now be written as:

$$\min_{\boldsymbol{\rho},\mathbf{a}} \max_{\mathbf{q}} E(\boldsymbol{\rho}, \mathbf{a}, \mathbf{q}) \tag{3.14}$$

where

$$E(\boldsymbol{\rho}, \mathbf{a}, \mathbf{q}) = \sum_{p \in \mathcal{P}} E_a(a_p, \rho_p) + \langle \boldsymbol{\nabla}_p^{\boldsymbol{\rho}}, \mathbf{q}_p \rangle - \delta_q(\frac{\mathbf{q}_p}{g_p}) - \frac{\epsilon}{2} \frac{\|\mathbf{q}_p\|_2^2}{g_p} \tag{3.15}$$

and

$$E_a(a_p, \rho_p) = \frac{1}{\lambda} E_\phi(a_p) + \frac{1}{2\theta} \|\rho_p - a_p\|_2^2 . \tag{3.16}$$

The energy in Equation (3.15) can be optimised by performing gradient ascent on dual variables $\mathbf{q}_p, p \in \mathcal{P}$, followed by gradient descent on primal variables $\rho_p, p \in \mathcal{P}$, and an exhaustive point-wise search in the discrete label space $\mathcal{L}$ for finding a_p that minimizes $E_a(a_p, \rho_p), p \in \mathcal{P}$ as in Chambolle and Pock [2010] and Newcombe et al. [2011]. This process is repeated iteratively while decreasing θ slowly. The variable updates within the primal and dual steps and the point-wise search can occur in parallel, and thus (3.15) can be optimized efficiently in GPU hardware. Using a sparse pairwise graph structure for the regularization term adds to the gradient computation efficiency.

Using the method in Newcombe et al. [2011] we compute the upper and lower bounds for the exhaustive search since the required search space in $\mathcal{L}$ grows narrower as θ decreases, and also perform a single Newton step on the optimal $\mathbf{a}$ at each time step for achieving sub-label accuracy (using numerical derivatives of $E_{\mathbf{a}}$ w.r.t $\mathbf{a}$ around its current discrete solution). Algorithm 3.1 summarizes the steps for obtaining the solution.

In Algorithm 3.1, ∇_p^q denotes the following operation:

$$\nabla_p^q = (q_{px}c_{pi} - q_{rx}c_{rr}) + (q_{py}c_{pj} - q_{sy}c_{ss}), \tag{3.17}$$

where r and s correspond to pixel locations neighbouring that of p in the *negative* x and y directions in the image plane. Note that $\boldsymbol{\nabla}_p^{\boldsymbol{\rho}}$ and ∇_p^q can be considered a generalization of the gradient and divergence operations. The dual and primal step sizes are denoted by σ_q and σ_ρ respectively.

3.3.4 Camera Tracking and Frame selection

We use the framework in ORB-SLAM (Mur-Artal et al. [2015]) for accurate feature-based camera tracking, and providing the required transformation T_{nr} for computing the photometric cost. The choice of $\mathbf{I}_r$ and its overlapping frame set greatly influence the reliability of the data term. We consider a pre-defined frame buffer window of size $N = N_p + N_f$ around $\mathbf{I}_r$, where N_p is the maximum number of past overlapping *keyframes* (large-baseline) to consider, and N_f is the maximum number of future overlapping *frames* (small-baseline) to consider. The past keyframes are selected with the help of ORB-SLAM's covisibility graph, and they are stored in a fixed length rolling buffer in GPU memory so that need not be re-copied. Setting the future frame count to 0 allows for "just-in-time" reconstruction as demonstrated in the video accompanying Figure 3.1, while increasing it correspondingly increases the mapping latency. Each keyframe change in the ORB-SLAM tracker (Mur-Artal et al. [2015]) sets

Algorithm 3.1: Optimisation procedure for solving for optimal inverse depth values $\rho_p = a_p, p \in \mathcal{P}$ for a keyframe $\mathbf{I}_r$

```
1  Initialize $q_p = 0, p \in \mathcal{P}$ ;
2  Initialize $a_p = \rho_p = \arg\min_{a_p \in \mathcal{L}} E_\phi(a_p), p \in \mathcal{P}$ ;
3  Initialize $\theta = \theta_{start}$;
4  Compute $g_p$, $c_{pq}$, and $c_{pp}, p \in \mathcal{P}, q \in \mathcal{N}(p)$;
5  repeat
6      $\mathbf{q}_p \Leftarrow (\mathbf{q}_p + \sigma_q \boldsymbol{\nabla}_p^\rho)/(g_p + \sigma_q \epsilon), p \in \mathcal{P}$ ;
7      $\mathbf{q}_p \Leftarrow g_p \mathbf{q}_p / \max(1, \|\mathbf{q}_p\|_2), p \in \mathcal{P}$ ;
8      $\rho_p \Leftarrow (\rho_p + \sigma_\rho(-\nabla_p^q + \frac{1}{\theta} a_p))/(1 + \frac{\sigma_\rho}{\theta}), p \in \mathcal{P}$ ;
9      Compute bounds for point-wise search ;
10     $a_p \Leftarrow \arg\min_{a_p \in \mathcal{L}} E_a(a_p, \rho_p), p \in \mathcal{P}$ ;
11     Do Newton-step on $a_p$ (if step-size < bin size) ;
12     Decrease $\theta$ ;
13 until convergence;
```

a flag that indicates sufficient motion to initialize a new $\mathbf{I}_r$. Once the flag is set, and the previous keyframe's cost volume update and inverse depth optimisation is complete, the current image in the sequence is set as $\mathbf{I}_r$. Figure 3.3 illustrates the data flow and steps undertaken during a single keyframe reconstruction.

3.3.5 Scaling Camera Translations

As an optional step we scale camera translations to a fixed scale to facilitate the choice of a consistent set of inverse depth labels $\mathcal{L}$. This is due to the inherent scale ambiguity in monocular SLAM which may require the inverse depth label range to be manually tuned every time the system is re-run. For automatically recovering the approximate scale we use absolute depth predictions from Eigen and Fergus [2015] which are predicted in parallel with surface normals and bear minimal overhead to prediction time (sub-section F). These depth predictions are able to provide a rough idea about the scene's scale, having learnt approximately the relationship between object features and their typical size. Note that we use depth predictions *only* for scale recovery and hence up-to-scale reconstructions are possible without it.

We do a 1-point RANSAC based least-square fit to find the approximate multiplicative scale factor that will align ORB-SLAM's sparse 3D map with the corresponding CNN depth predictions for the current keyframe. This scale is then used to normalize the camera translations in T_{nr}, which in turn enables the use of a fixed set of inverse depth labels (based on metric units) for the cost volume. We perform a running average of the scale factors recovered for each $\mathbf{I}_r$ to improve the reliability of the scale factor estimate.

3.3.6 Surface Normals Prediction

For regressing surface normals directly from the keyframe image, we utilize the multi-scale CNN model proposed in Eigen and Fergus [2015]. We use their VGG model variant with VGG-16-based convolutional layers in scale 1 followed by 2 fully connected layers. The input RGB image to the

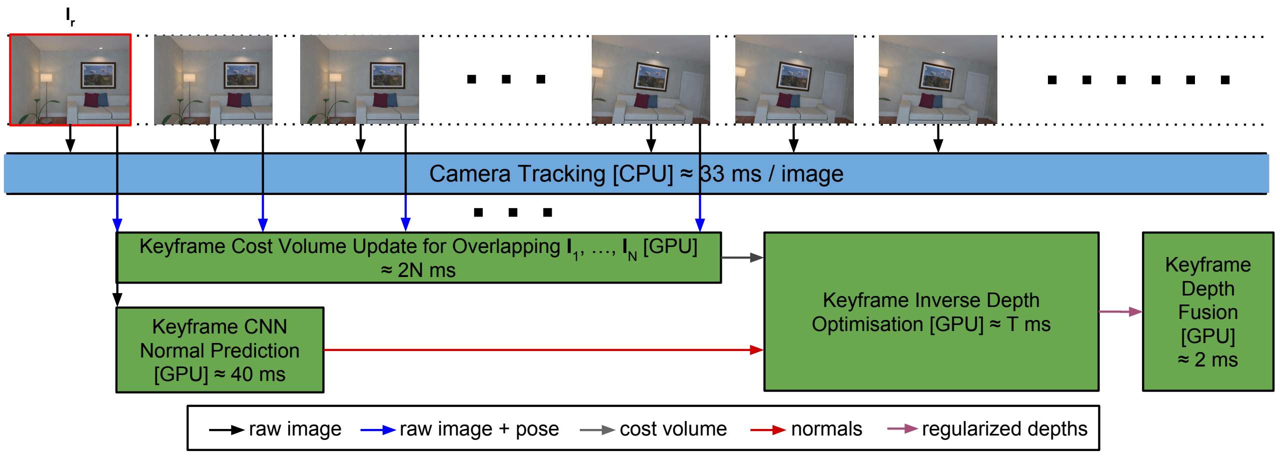

I_r
Camera Tracking [CPU] ≈ 33 ms / image
Keyframe Cost Volume Update for Overlapping I_1, ..., I_N [GPU] ≈ 2N ms
Keyframe CNN Normal Prediction [GPU] ≈ 40 ms
Keyframe Inverse Depth Optimisation [GPU] ≈ T ms
Keyframe Depth Fusion [GPU] ≈ 2 ms
raw image
raw image + pose
cost volume
normals
regularized depths

Figure 3.3: Key components of the framework and data flow. The depicted steps are repeated for each keyframe $\mathbf{I}_r$. Our GPU implementation is based on CUDA. Camera tracking (Mur-Artal et al. [2015]) runs in the main CPU thread. The cost volume updates and CNN normal predictions both run on the GPU in two independent CUDA streams, and are managed in parallel with the main thread. Optimisation and depth map fusion run subsequently on the GPU and are also managed in parallel with the main thread. Similar to Newcombe et al. [2011] the optimisation time T is dependent on the optimisation parameters like step sizes and θ scheduling policy, and is $\approx$ 50 ms/keyframe for fast but less accurate reconstructions, and $\approx$ 800 ms/keyframe for more accurate but higher latency reconstructions. Mapping latency (time between successive keyframe reconstructions) is $\approx$ $(33 + max(40, max(33N_f, 2(N-1)) + 2) + T + 2)$ ms where $N = N_p + N_f$ is the total number of images overlapping $\mathbf{I}_r$. If the number of future frames N_f is set to 0, minimum mapping latency for reasonable accuracy is $\approx$ 150 ms in practice. Frequency of keyframe reconstruction also depends on amount of camera translation which determine when a keyframe change should occur.

network is first resized to 320x240 and then centre cropped to 304x228. The cropping is required as the network was trained with randomly cropped images at that same crop resolution. Scale 1 mainly operates on a coarser image resolution and extracts more global features. Scales 2 and 3 consist of fully convolutional layers which operate on fine and finer image resolutions respectively and extract more local features. Scales 2 and 3 receive upsampled output from the preceding coarser scales (Eigen and Fergus [2015]).

The network at scale 3 simultaneously regresses a surface normal map and depth map for the input keyframe image at 147x109 resolution. In spite of the low-resolution output a major portion of the scene detail is still captured. We bilinearly upsample the predictions by a factor of 2 to the corresponding region in the 320x240 input image. The small amount of missing information at the border of the resulting normal/depth map is due to effects of cropping at the input and intermediate layers in the network. We do not perform inverse depth regularization in this border region.

We replicated Eigen and Fergus [2015]'s model in the efficient Caffe (Jia et al. [2014]) framework and transferred the learnt weights. Their model has been trained on millions of indoor images (data augmentation included) in the training set of the raw NYU-Depth V2 dataset (Silberman et al. [2012]). The combined prediction time for a surface normal map and depth map in Caffe is $\approx 40ms$ in GPU mode.

3.3.7 Volumetric Fusion

As a post-processing step, the depth maps resulting from the optimisation are fused into a global volumetric model based on truncated signed distance function, using the open-source InfiniTAM system (Prisacariu et al. [2014]). The overall framework is summarized in Figure 3.3.

3.4 Experimental Results

Using the smoothness regularizer ($\gamma = 1$ in Equation (3.11)) as the baseline, we explore the improvements after using the surface normals regularizer ($\gamma = 0$) on a large number of sequences in several video datasets. All our experiments are conducted on a standard desktop PC with an Intel i7 4790 CPU and a Nvidia GTX 980 4GB GPU. We also compare against single-view CNN depth predictions from Eigen and Fergus [2015] to observe how well a pure-learning based approach compares with our combined learned prior and photometric error-based approach. Note however that this is not a fair comparison for single-view depth estimation as it doesn't take into account photo-consistency information unlike the other two methods. Nevertheless it still serves as an indicator as to what error and accuracy measures single-view depth predictors tend to suffer most from compared to our results.

In our experiments we bilinearly downsample input images to 320x240 to ensure low-latency fusions and smooth operation for the entire pipeline. This is at a slight compromise of data-term quality. The neural network also takes in input at this resolution which made the choice more appropriate. Given hardware speed and memory constraints, the reduced resolution also allowed the use of a relatively large label set $\mathcal{L}$ for the cost volume. We use a fixed inverse depth range of 0 to 4 with 256 bins, sufficient for reconstructing small to large scale environments. Note that no finetuning of the network is done on any of the sequences used for evaluation. This allows us to test the generalization capabilities

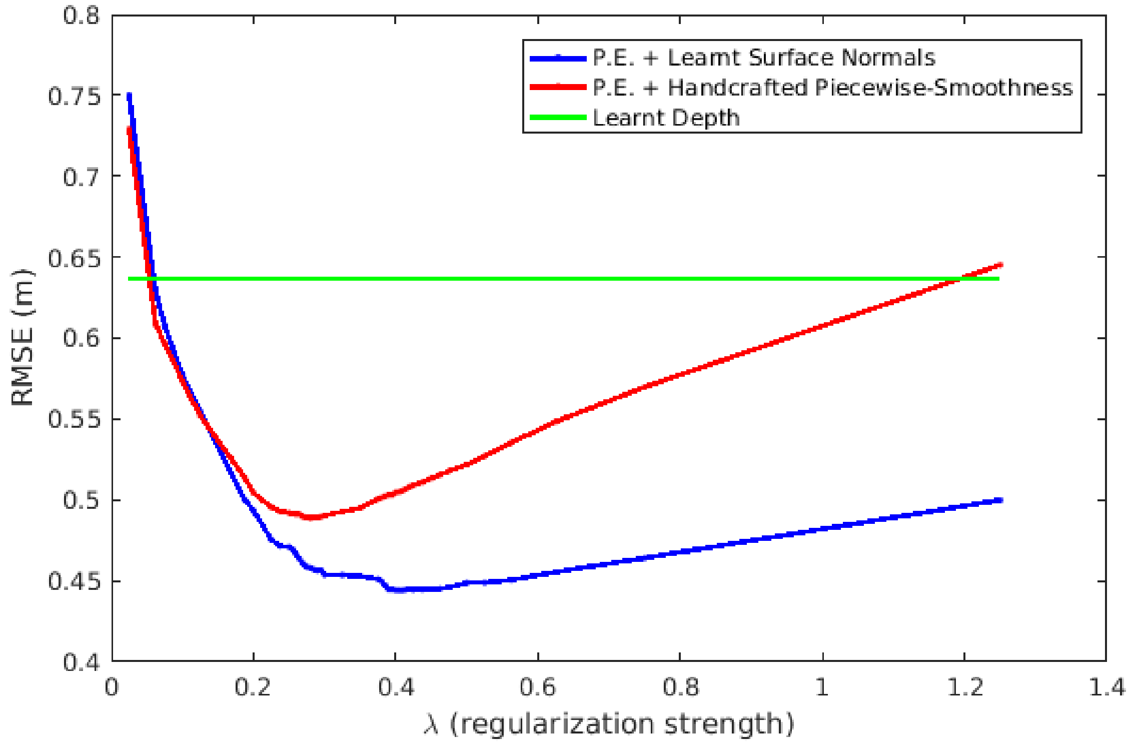


Figure 3.4: Average keyframe RMS reconstruction error (m) w.r.t regularization strength (λ) on the raw NYU-D V2 test sequences. Notice that the graph for the normal-based regularizer achieves a lower RMSE and its error remains lower after a certain threshold, implying that it is less sensitive to the choice of regularization strength.

of the neural network.

We performed quantitative comparisons using the RGB-D tracking feature in Mur-Artal et al. [2015] as (i) it allowed for ease of repeatability of experiments (more consistent camera pose estimates and keyframes selected, no scale ambiguity) and (ii) it leaves out errors in camera poses when comparing the two regularization methods. The qualitative results were generated using regular monocular tracking.

We first tune λ for the two regularizer types using test scenes in the raw NYU-D V2 dataset, so that the optimal λ can be chosen. Following Eigen and Fergus [2015] we split the raw NYU-Depth dataset by Silberman et al. [2012] into test and train scenes based on the official dataset split, using scenes that don't contain train images for testing. This gives 247 different indoor test sequences out

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CNN Depth	0.637	0.226	0.163	0.135	0.738	0.937	0.982
P.E. + Smoothness	0.522	0.206	0.123	0.111	0.834	0.949	0.979
P.E. + Normals	**0.449**	**0.174**	**0.086**	**0.076**	**0.893**	**0.964**	**0.985**

Table 3.1: Quantitative results on 25 raw NYU-D V2 Dataset test sequences. P.E. = Photometric Error. CNN Depth is the output of Eigen and Fergus [2015]. The average errors and accuracy are for keyframe reconstructions against Kinect depth maps (where valid depths are available). The results here are shown for the optimal lambda values for normals and smoothness regularizer.

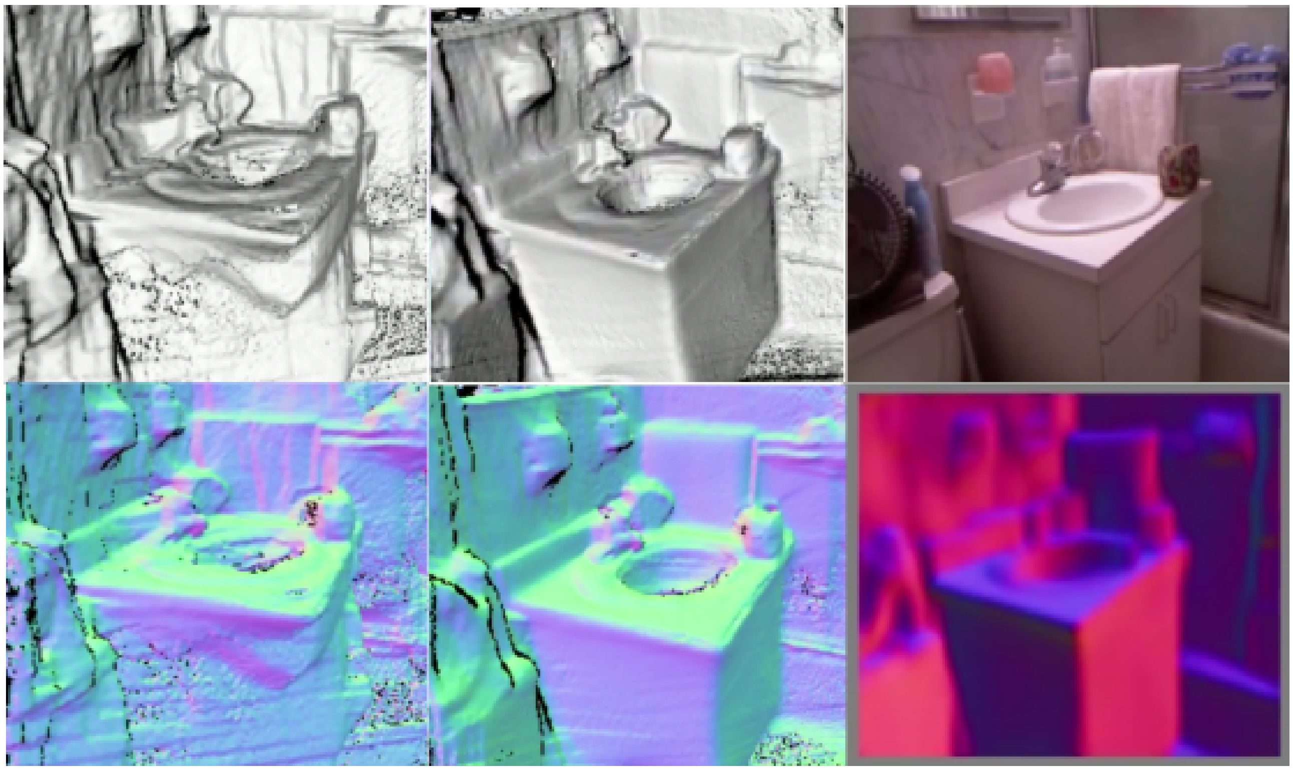


Figure 3.5: Qualitative results on NYU raw dataset 'bathroom_0003' test sequence. Phong shaded fused reconstruction using smoothness prior (top left), phong shaded fused reconstruction using normals prior (top middle), a rgb keyframe image in the sequence (top right), surface normal rendering of fused smoothness-prior reconstruction (bottom left), surface normal rendering of fused normal-prior reconstruction (bottom middle), corresponding normal predictions for rgb keyframe image (bottom right). Note the more accurate reconstruction of textureless regions like the inside of the round sink using the normal-prior. A live comparison video is available at https://youtu.be/BRLN-1MTZtw.

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CNN Depth	1.141	0.368	0.227	0.261	0.543	0.820	0.923
P.E. + Smoothness	0.563	**0.215**	0.106	0.091	0.854	**0.924**	**0.973**
P.E. + Normals	**0.558**	**0.215**	**0.103**	**0.089**	**0.863**	0.922	0.969

Table 3.2: Quantitative results on the TUM dataset 'fr2_desk' sequence. P.E. = Photometric Error. CNN Depth is the output of Eigen and Fergus [2015].

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CNN Depth	0.829	0.426	0.295	0.261	0.472	0.781	0.905
P.E. + Smoothness	0.287	0.118	0.062	0.041	0.943	0.988	0.997
P.E. + Normals	**0.214**	**0.094**	**0.047**	**0.022**	**0.963**	**0.993**	**0.998**

Table 3.3: Quantitative results on ICL-NUIM dataset 'lr kt0' sequence. P.E. = Photometric Error. CNN Depth is the output of Eigen and Fergus [2015].

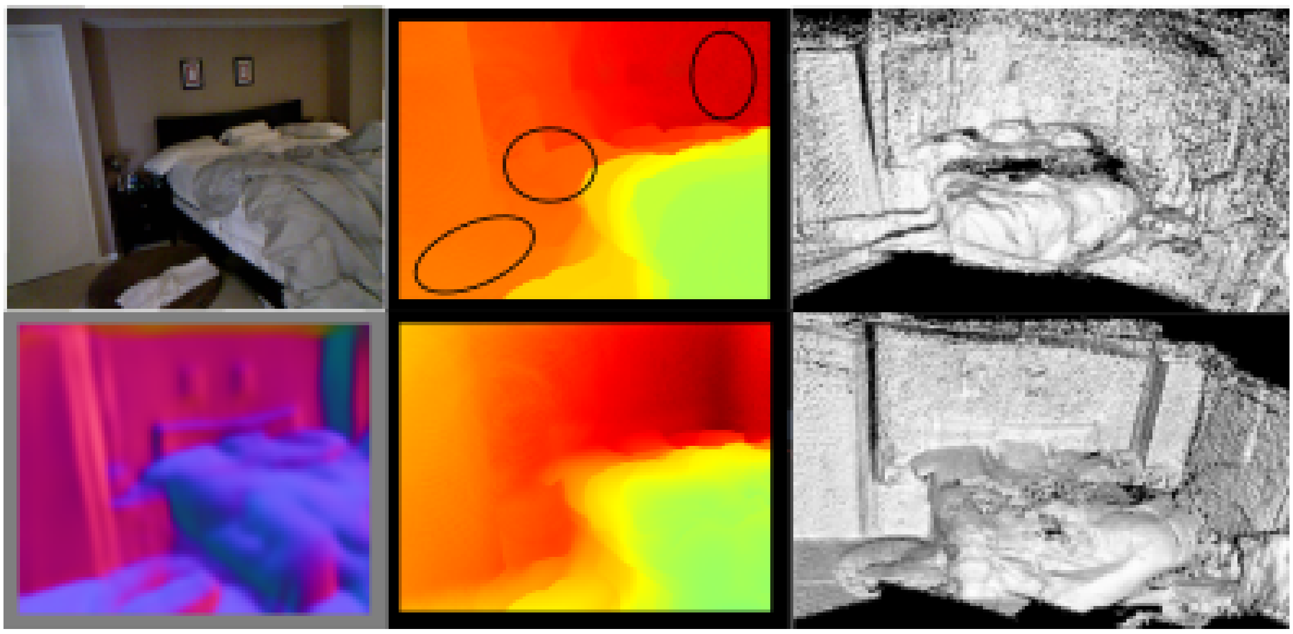


Figure 3.6: Qualitative results on NYU raw dataset 'bedroom_0048' test sequence. Input rgb image (top left), reconstructed keyframe depth map with smoothness regularizer (top middle), fused reconstruction using smoothness regularizer (top right), keyframe surface normal prediction (bottom left), reconstructed keyframe depth map with normal-based regularizer (bottom middle), fused reconstruction using normal-based regularizer (bottom right). Note the more accurate reconstruction of the wall and floor overall when using the normal prior (which is more apparent in the phong-shaded reconstruction).

of which we choose 25 by sampling every tenth test sequence (when the scenes are sorted by scene name in alphabetical order) to roughly correspond to all the different types of indoor scene categories present in the raw test set. The plot of RMS error vs λ for the two regularizers are provided in Figure 3.4. It can be seen that as we vary λ, the error compared to the ground-truth remains lower for the normals-based regularization. This is because even though the regularization strength is high, the normals guide the depths to the right solution more accurately, while with the smoothness regularizer, a higher strength causes piecewise planar reconstructions that are fronto-parallel to the keyframe image plane. This effect is more apparent in areas where there is little texture as expected.

Quantitative results on these sequences for the following error and accuracy measures are given in Table 3.1. Note that d_p and d_p^{gt} denote regularized depth and groundtruth depth respectively of a pixel location corresponding to p.

rms: $\sqrt{\frac{1}{|P|}\sum_{p\in P}\|d_p - d_p^{gt}\|^2}$

log rms : $\sqrt{\frac{1}{|P|}\sum_{p\in P}\|log(d_p) - log(d_p^{gt})\|^2}$

abs. rel: $\frac{1}{|P|}\sum_{p\in P}\frac{|d_p - d_p^{gt}|}{d_p^{gt}}$

sq. rel: $\frac{1}{|P|}\sum_{p\in P}\frac{\|d_p - d_p^{gt}\|^2}{d_p^{gt}}$

Accuracies: % of d_p s.t. $max(\frac{d_p}{d_p^{gt}}, \frac{d_p^{gt}}{d_p}) = \delta < thr$

The errors are computed at locations where both Kinect raw depth data is available and where

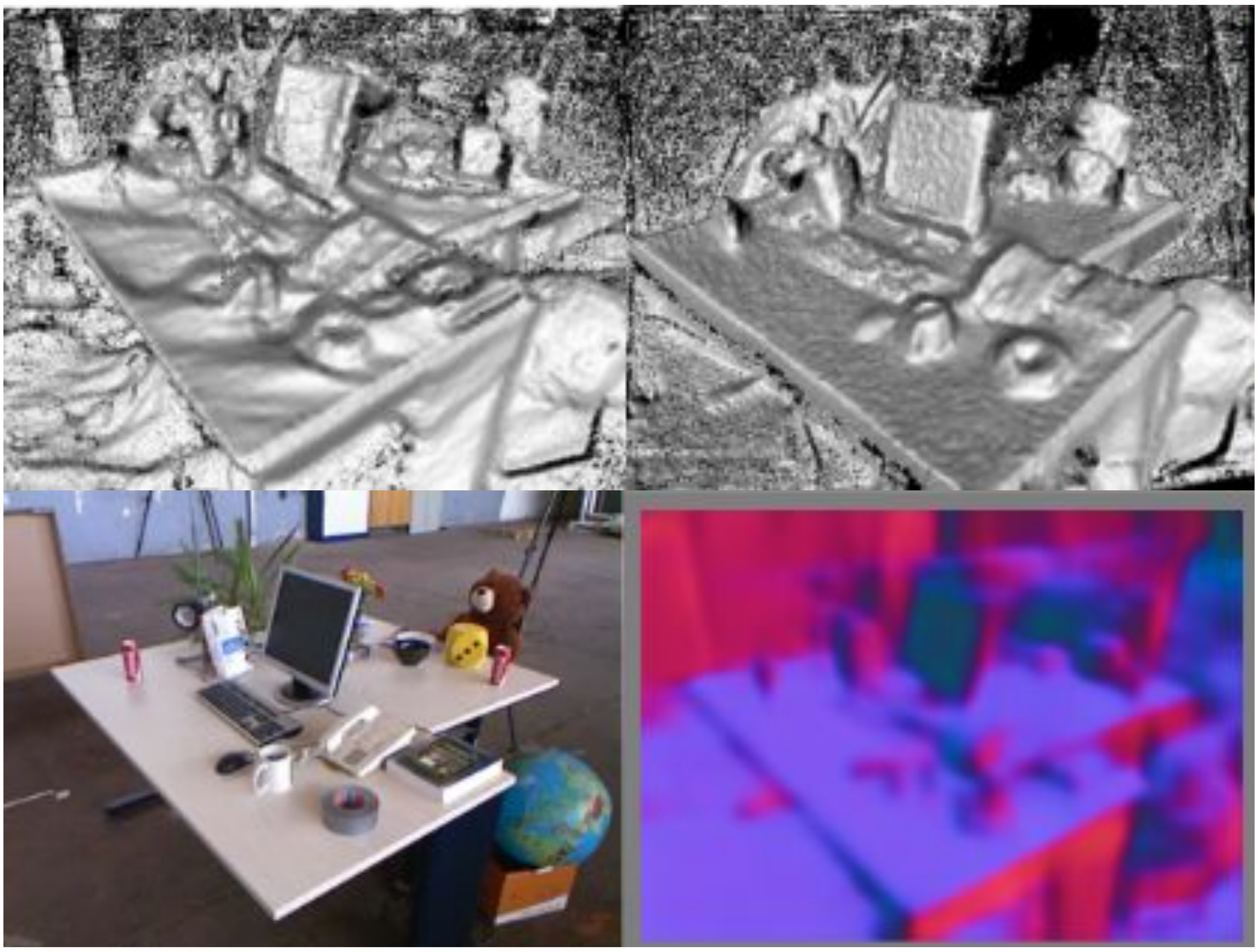


Figure 3.7: Qualitative results on TUM dataset 'fr2_desk' sequence. From left-to-right are fused reconstruction using smoothness prior, fused reconstruction using normals prior, a rgb keyframe image in the sequence, and corresponding normal predictions for rgb keyframe image.

depth regularization is performed (regions excluding the small border where predictions are not made). The regularized depth maps and CNN depth predictions are bilinearly upsampled to 640x480 resolution prior to evaluating against the raw Kinect depth maps. Note that the same optimisation and cost-volume-related parameters (30 past and 30 future live frames overlapping a keyframe) were used for comparing the two regularizer types. We follow the same θ scheduling policy as Newcombe et al. [2011] with similar choice of parameters. The table, in particular the low threshold accuracy column, help validate that the normal-prior helps in recovering the fine details in the scene. Qualitative comparisons are shown for two NYUv2 raw test sequences in Figures 3.5 and 3.6. The improvements in reconstruction in terms of both fine detail and global scene structure are apparent, especially in textureless regions.

The same experiments were carried out on the TUM dataset (Sturm et al. [2012]) and the living room sequence 'lr kt0' in the ICL-NUIM dataset (Handa et al. [2014]). Quantitative results for these sequences are shown in Tables 3.2 and 3.3, and the qualitative result for the TUM sequence is shown in Figure 3.7. Again a similar trend to that observed before can be seen. The high accuracy measure for the TUM sequence favour the learned normals prior, and this is evident in Figure 3.7 by the ability of the normal prior to respect the true surface structure of the scene, especially for closer regions to the camera. However, the smoothness prior scores slightly better in the low accuracy measure, indicating that the normal-based reconstructions contain slightly more outliers.

While this is yet to be confirmed through experimentation, this result could be due to 1) inaccuracies in normal predictions for far points in particular, or 2) optimization settings being non-optimal for the normal-based prior resulting in the solution being trapped in a local optimum (note that we use identical settings for optimizing both the normal and smoothness prior models for fair comparison). Nevertheless, we revisit this experiment in Chapter 4, where it was seen that a combination of good features for matching and our learned normal prior model gave the best result overall. Based on all three tables we can also observe that the CNN depth predictions (at least those by Eigen and Fergus [2015]) do not generalize to new scene types (outside the NYU dataset) as well as the other two methods.

We also have tried manually scaling the depth maps by the ratio of train-test scene focal length, however this did not show improvement in evaluation results. The experiments in CNN-SLAM (Tateno et al. [2017]) suggest that this depth scaling helps at least when used to support camera tracking. A straightforward approach to improve upon learned depth maps from a single-view based CNN is to fuse its output depth maps across multiple views based on their relative poses. During the initial stages of our work we fused single-view depth map predictions from the VGG-F based model in Liu et al. [2015b], which are refined with a CRF post-processing step. The results obtained by simply fusing depth maps however were less than satisfactory even with groundtruth poses. Fusing the depth predictions by Eigen and Fergus [2015] on the other hand produced more coherent results (top-right result of Figure 3.8). This showcases the obvious and crucial dependency of the accuracy and generalizability of the depth predictor for multi-view depth fusion. However, as seen in Figure 3.8, the result of simple depth fusion is inferior to that of our proposed method combining learned normals with photo-consistency.

It is interesting to re-visit this simple fusion strategy once (and if at all) single-view depth predictors get to the point where they are consistently closer to groundtruth than they are now, more importantly,

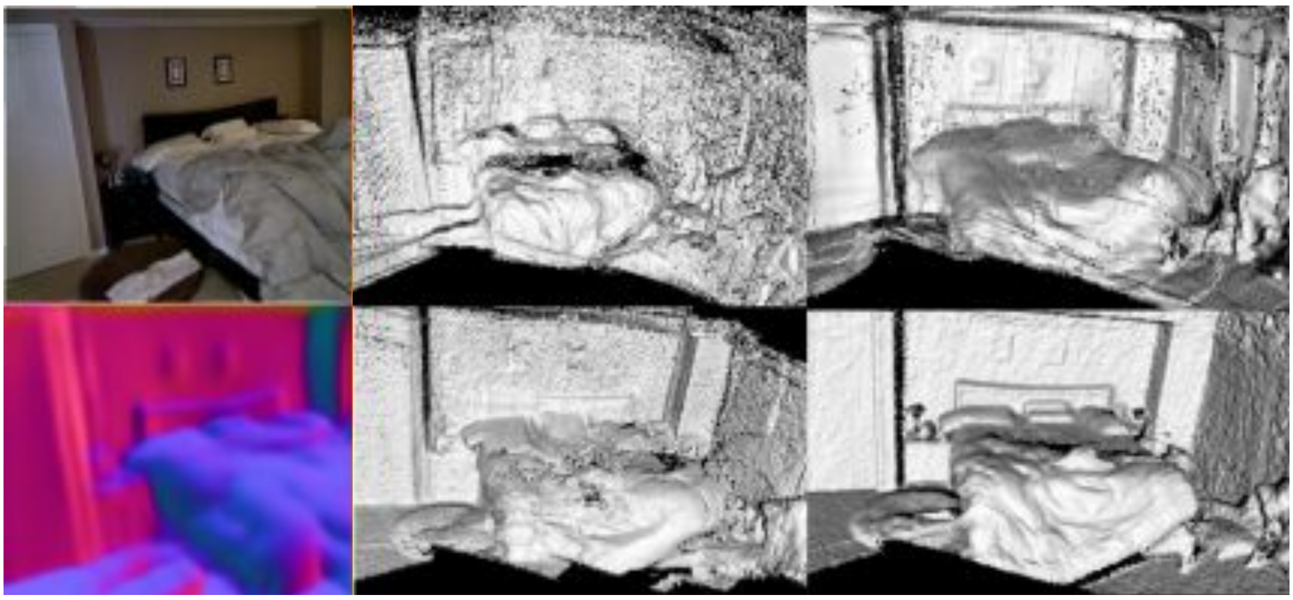


Figure 3.8: Dense volumetric reconstruction results of fusing single view depth predictions, and comparison with other methods. In the top row, left-to-right: an image in sequence, result of fusing depth maps from DTAM (with the smoothness prior replacing our normal prior), result of fusing depth maps from Eigen and Fergus [2015]; In the bottom row, left-to-right: corresponding predicted surface normal map for the image using Eigen and Fergus [2015], the result of fusing depth maps from our proposed method which incorporates our learned normal-prior, and the result of fusing Kinect depth maps. Note that InfiniTAM (Prisacariu et al. [2014]) was used for fusion in all cases. While fusing learned depth maps from Eigen and Fergus [2015] produce rather coherent results (at least in this NYUv2 test sequence), our result which combines learned normals with photo-consistency produces the closest result to that of KinectFusion (which serves as a reference).

in previously unseen environments. Nevertheless, as seen in Figure 3.9, a learned depth map still contains information complementary to a surface normal map in terms of *depth relationships*, especially at depth discontinuities. Furthermore, as discussed earlier, surface normals are likely to be unreliable when neighboring points in 3D are spread further apart, thus violating the local planarity assumption. Our formulation in Chapter 5 takes advantage of these learned depth relationships for “just-in-time” reconstruction. At the end of Chapter 5, we also propose a potential unified formulation to exploit depth priors *jointly* with normal priors and photo(or feature)-consistency among other learned information, for dense keyframe reconstruction.

3.5 Future Directions

3.5.1 Outdoor Experiments

While our experiments were limited to reconstructing indoor environments, the same framework in theory can be used for building dense maps of outdoor scenes, given the large depth range covered by the cost volume and large-scale volumetric fusion capabilities of InfiniTAM (Prisacariu et al. [2014]). Since it is common to use different models for indoor and outdoor scenes for best results, it will be required to re-train a CNN for surface normal prediction in outdoor scenes. It would be also of interest to implicitly learn a common model to predict normals in both indoor and outdoor settings, or

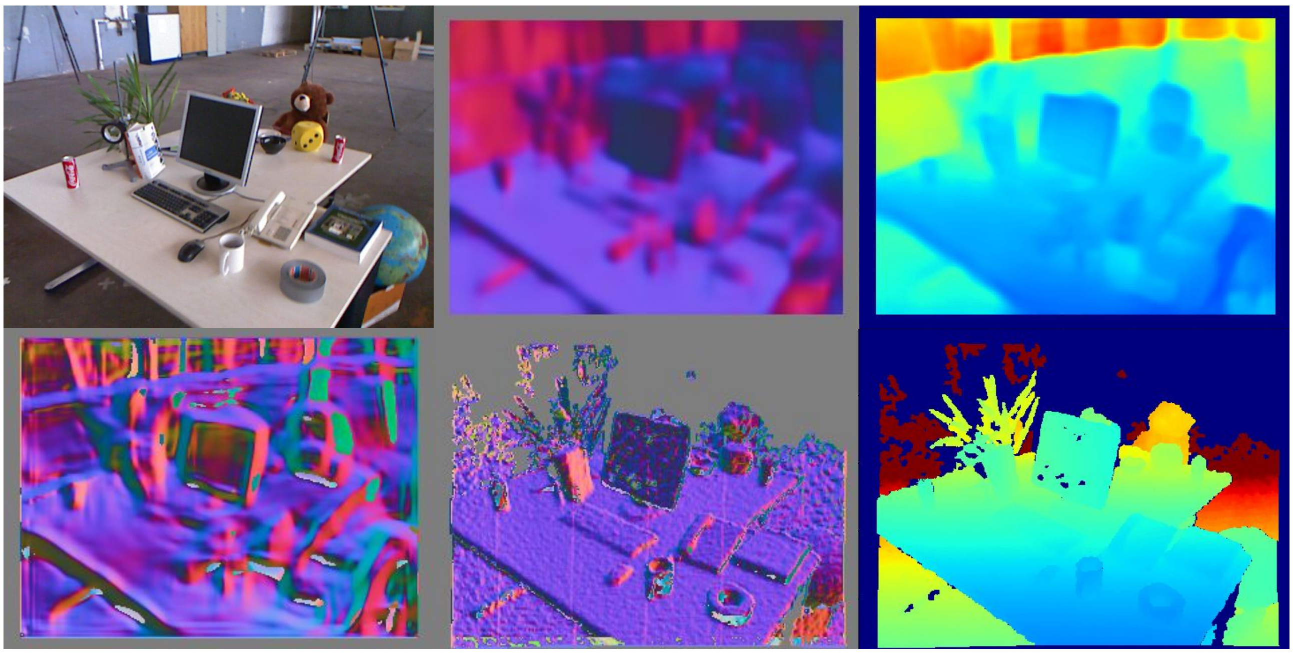


Figure 3.9: The figure illustrates the complementary information present in both depth and normal predictions. First row from left: An input image, predicted surface normal (Eigen and Fergus [2015]), and predicted depth map (Eigen and Fergus [2015]) from the same network. Second row from left: surface normals computed from predicted depth map, groundtruth surface normals computed from groundtruth depth map, groundtruth depth map. Notice that the surface normals computed from the predicted depth map are inconsistent with the more accurate normals of the predicted surface normal map. Therefore the predicted surface normal map has complementary information to improve the predicted depth map, which the network of Eigen and Fergus [2015] has failed to learn implicitly. On the other hand, the predicted depth map has richer information about depth relationships at depth discontinuities than the predicted surface normal map, although fairly inaccurate in terms of absolute depth value (its color mapping is set to be consistent with the groundtruth depth map). At the end of Chapter 5 we look at a potential way to incorporate both these learned depth and normal priors into our energy formulation for keyframe reconstruction.

explicitly learn to select the indoor/outdoor CNN model based on the input data. The main difficulty in training for normal prediction outdoors is in acquiring densely labeled outdoor depth maps (required for generating ground truth normals) for training.

However, recent unsupervised learning scheme similar to Garg et al. [2016]; Godard et al. [2017]; Yang et al. [2017] can be used to train for dense depth prediction in outdoor scenes, and normals can be computed from the output of such a network. In the case where sparse LIDAR depth maps are available, a denser depth map for normal computation can be generated using an image-guided depth inpainting method like cross-bilateral filtering (Petschnigg et al. [2004]) and colorization (Levin et al. [2004]), or our proposed method in Chapter 5 can be used in combination with learned depths for more reliable inpainting.

We can either compute normals from the depths maps at test time, or more ideally we can train a separate network to directly predict normals to match the computed normals from the depth maps, same as our indoor experiments. This is because as we saw in Figure 3.9, a learned surface normal map provides complementary information to a learned depth map, and a surface normal map computed from a learned depth map may be less accurate than an independently predicted surface normal map. In Section 3.5.3, we also discuss how we could potentially learn to predict normals through minimization of a loss function that doesn't involve explicit computation of normals, paving the way for joint-unsupervised learning of depth and normals.

3.5.2 Learning Surface Normal Prior Confidences

It is common to use the handcrafted image-edge weight ($g_p = e^{-\alpha|\nabla I_r(u_p)|_2^\beta}$) as a heuristic to reduce penalization of the regularization term at depth discontinuities where the local planarity assumption does not hold. However not all image edges correspond to depth discontinuities, and therefore the contribution of the normal prior may be suppressed unnecessarily. Moreover this weight does not account for the uncertainties of the predicted normals.

Here we propose a formulation to learn to predict pixelwise confidence weights from the image via a CNN, such that these weights are more reflective of the confidences of the normal priors. We again focus on our proposed normal prior term which is based on the relationship between a pair of 3D points and its corresponding predicted surface normal $\hat{n}_i$ in the camera coordinates of the reference image $\mathbf{I}_r$:

$$\begin{aligned} \langle \hat{n}_i, d_j\tilde{x}_j - d_i\tilde{x}_i\rangle &= 0 \\ d_j\langle \hat{n}_i, \tilde{x}_j\rangle - d_i\langle \hat{n}_i, \tilde{x}_i\rangle &= 0 \\ \rho_i\langle \hat{n}_i, \tilde{x}_j\rangle - \rho_j\langle \hat{n}_i, \tilde{x}_i\rangle &= 0 \end{aligned} \tag{3.18}$$

Thus we can formulate the following objective:

$$\min_{\sigma_{ij}} \sum_{i,j\in\mathcal{N}(i)} -ln\frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp -\frac{1}{2\sigma_{ij}^2}(\rho_i\langle \hat{n}_i, \tilde{x}_j\rangle - \rho_j\langle \hat{n}_i, \tilde{x}_i\rangle)^2 \tag{3.19}$$

Let us write $c_i^{\hat{n}} = 1/\sigma_{ij}$ to represent the confidence of the predicted surface normal $\hat{n}_i$. With this parameterization, Objective (3.19) can be written as:

$$\min_{c_i^{\hat{n}}} \sum_{i,j \in \mathcal{N}(i)} \frac{(c_i^{\hat{n}})^2}{2} (\rho_i \langle \hat{n}_i, \tilde{x}_j \rangle - \rho_j \langle \hat{n}_i, \tilde{x}_i \rangle)^2 - \frac{1}{2} \sum_{i,j \in \mathcal{N}(i)} ln \frac{(c_i^{\hat{n}})^2}{2\pi} \tag{3.20}$$

Simplifying the above expression and dividing by the neighborhood size $|\mathcal{N}(i)|$ leads to the following:

$$\min_{c^{\hat{n}}} \frac{1}{|\mathcal{N}|} \sum_{i,j \in \mathcal{N}(i)} \frac{(c_i^{\hat{n}})^2}{2} (\rho_i \langle \hat{n}_i, \tilde{x}_j \rangle - \rho_j \langle \hat{n}_i, \tilde{x}_i \rangle)^2 - \sum_i ln c_i^{\hat{n}} + K \tag{3.21}$$

where $K = -\frac{1}{2|\mathcal{N}(i)|} \sum_{i,j \in \mathcal{N}(i)} ln \frac{1}{2\pi}$.

Gradient expressions for Objective (3.21) with respect to $c_i^{\hat{n}}$ can be easily derived. We can now solve for Objective (3.21) (but summed over all training instances) via Stochastic Gradient Descent (SGD), with $lnc_i^{\hat{n}}$ regressed at the network output. Note that regressing for $lnc_i^{\hat{n}}$ instead of $c_i^{\hat{n}}$ is more desirable as it prevents a negative value being predicted for $c_i^{\hat{n}}$ and the output as well as the neural network weights will be in a more manageable range.

Normalizing $c^{\hat{n}}$: The range of values of the learned confidence map $c^{\hat{n}}$ may sometimes be very large depending on the nature of the normal inconsistency errors. After the CNN confidence predictor is trained we can simply normalize $c^{\hat{n}}$ by dividing it with a global constant such that it is in a manageable range. For instance the normalization could be done by: $c_i^{\hat{n}} \leftarrow c_i^{\hat{n}} / \sigma_{c_i^{\hat{n}}}$, where $\sigma_{c_i^{\hat{n}}}$ is the standard deviation of all the predicted $c_i^{\hat{n}}$s in the training set. This would make it easier to tune the hyper-parameter λ of our energy formulation for mapping during inference.

Dealing With Depth Discontinuities: Since our normal priors do not model depth discontinuities the point pairs involved at these regions should ideally be masked out. However, even without masking, the network should be able to implicitly learn to predict a very low confidence for these terms as the errors would most often be large. In the case we are provided with $b^{\hat{n}}$, a groundtruth depth discontinuity mask (or one that we have explicitly trained a CNN to predict given the ground truth mask), we can take this into account in our objective as follows:

$$\min_{c^{\hat{n}}} \frac{1}{|\mathcal{N}(i)|} \sum_{i,j \in \mathcal{N}(i)} \frac{b_i^{\hat{n}} (c_i^{\hat{n}})^2}{2} (\rho_i \langle \hat{n}_i, \tilde{x}_j \rangle - \rho_j \langle \hat{n}_i, \tilde{x}_i \rangle)^2 - \sum_i b_i^{\hat{n}} ln c_i^{\hat{n}} + K \tag{3.22}$$

where $K = -\frac{1}{2|\mathcal{N}(i)|} \sum_{i,j \in \mathcal{N}(i)} b_i^{\hat{n}} ln \frac{1}{2\pi}$.

3.5.3 Implicit Learning of Surface Normals

We can also learn to predict surface normals $\hat{n}$ from an image that minimize Objective (3.22). However two issues may arise: (1) The horizontal and vertical difference vectors have different magnitudes. Note that a surface normal requires at least two local difference vectors in 3D (e.g. horizontal and vertical) to be fully defined. In order to eliminate any bias towards a particular direction, these difference vectors need to be normalized so that their magnitudes are the same. Table 3.4 provides an empirical example showing why this is the case. (2) There are two surface normal solutions (pointing in opposite directions) that satisfy Objective (3.22). Measures need to be taken to make sure that the predicted normal is pointing towards the camera.

While it may not be mandatory, we can address the first issue by normalizing the magnitude of the difference vectors in Objective (3.22):

$$\min_{c,\hat{n}} \frac{1}{|\mathcal{N}(i)|} \sum_{i,j \in \mathcal{N}(i)} \frac{b_i^{\hat{n}} c_i^2}{2} \Big(\langle \hat{n}_i, \frac{d_j \tilde{x}_j - d_i \tilde{x}_i}{|d_j \tilde{x}_j - d_i \tilde{x}_i|_2} \rangle \Big)^2 - \sum_i b_i^{\hat{n}} ln c_i + K \tag{3.23}$$

where $K = -\frac{1}{2|\mathcal{N}(i)|} \sum_{i,j \in \mathcal{N}(i)} b_i^{\hat{n}} ln \frac{1}{2\pi}$.

Note that c_i here is not the same as $c^{\hat{n}}$ since the model in Objective (3.23) is different to our previous model in Objective (3.22) which did not normalize the magnitude of difference vectors to 1 and used inverse depth parameterization instead. Therefore $c_i^{\hat{n}} = c_i d_i d_j / |d_j \tilde{x}_j - d_i \tilde{x}_i|_2$. In this case c_i merely acts as a form of attenuation to implicitly capture the non-ideal properties of the model (e.g. local planarity assumption being weaker for point pairs that are further away, etc.) and learn better normals. This confidence weighting strategy has been shown to slightly improve the learning of depth maps and semantic label maps in Kendall and Gal [2017].

While not straightforward, the second potential issue may be addressed for instance by introducing a minus exponential layer as an activation function for the z component of the predicted surface normals to make sure that the predicted normals are all pointing towards the camera ($z < 0$).

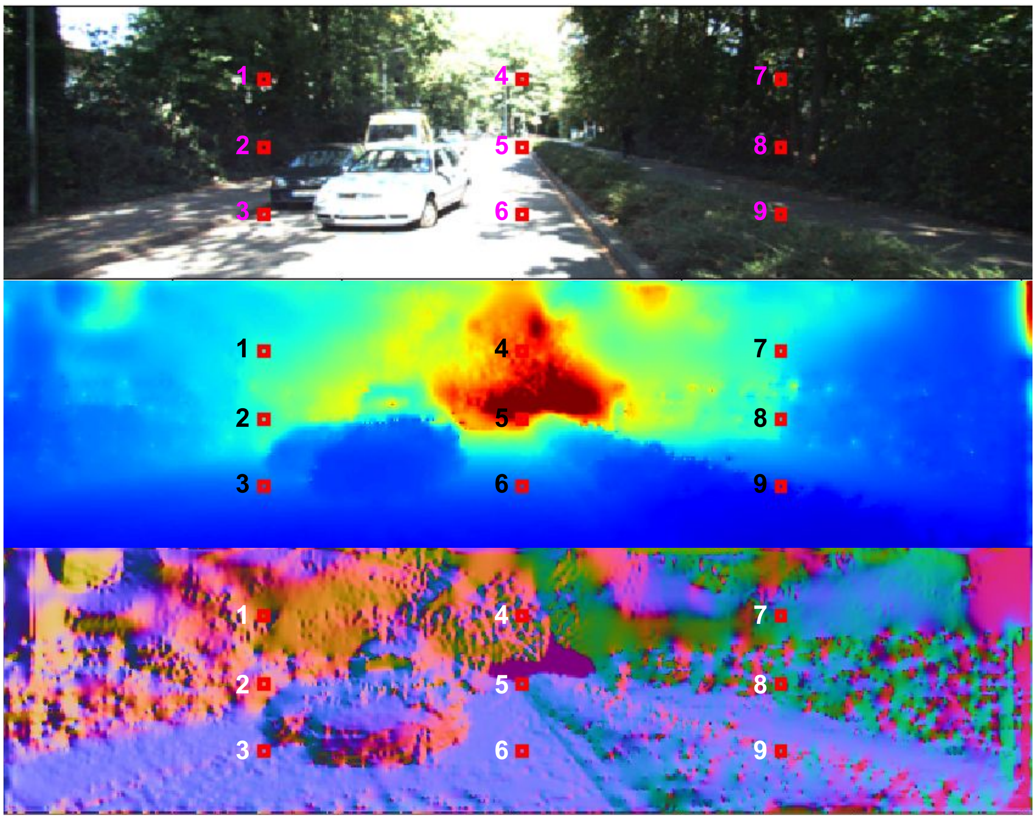
The benefit of our proposed formulation for learning normals is that one does not need to explicitly compute groundtruth normals based on the groundtruth depth map, but let the network implicitly regress for the surface normals that satisfy the local 3D surface orientations as specified by the groundtruth depth map.

Alternatively, one may use the cross product operation on the difference vectors in order to explicitly obtain the groundtruth surface normals and then use other losses commonly employed for normal prediction, such as the negative dot product loss between predicted and groundtruth normals as used in Eigen and Fergus [2015], or the L2 distance loss between predicted and groundtruth normals.

In the unsupervised learning scenario, as groundtruth depth is unavailable, the predicted depth map can be used as a proxy groundtruth for surface normal prediction. Our proposed inverse depth-normals prior can be then used in turn to regularize the depth network instead of the inverse depth smoothness prior as commonly employed in existing unsupervised learning methods (Garg et al. [2016]). Furthermore, a smoothness prior on normals that enforce nearby normals in the image to have the same orientation will also be necessary to regularize the normal prediction network. Note that both our inverse depth-normals prior and the normal smoothness prior are more realistic priors than the inverse depth smoothness prior that they replace.

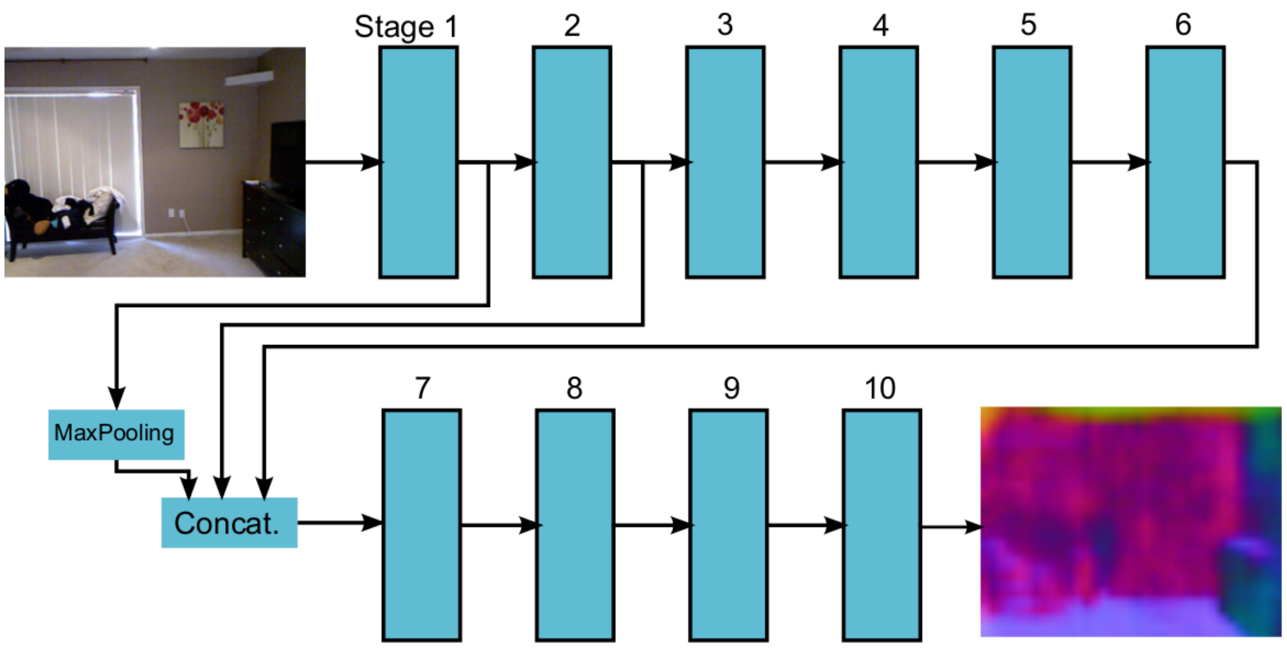
3.5.4 More Efficient Surface Normal Estimation

It is desirable to explore more efficient CNNs for surface normal prediction, for instance based on the more recent network architectures such as ResNet (He et al. [2016]), DenseNet (Huang et al. [2017]) and RefineNet (Lin et al. [2017]). Architectures for surface normal prediction like that of Eigen and Fergus [2015] and Bansal et al. [2016] demonstrate that surface normals are more accurate when low and high-level information are explicitly combined. However, it would be interesting to further investigate as to which layers in the network add to the accuracy of normal prediction so that unwanted filters can be removed, hence improving efficiency. This will also help us to better interpret the answers to questions such as *what high-level information, if at all, is implicitly accounted for in the*



Pixel	[x,y]	Depth (Center) m	Depth (Right) m	Depth (Bottom) m	Maximum Horizontal Penalty (10^{-4})	Maximum Vertical Penalty (10^{-4})
1	[152 , 40]	19.4	19.4	19.5	0.163	0.371
2	[152 , 80]	19.1	19.3	19.2	0.454	0.195
3	[152 , 120]	12.1	12.1	11.9	0.324	1.283
4	[304 , 40]	41.5	41.6	42.0	0.071	0.257
5	[304 , 80]	50.5	50.5	50.1	0.056	0.157
6	[304 , 120]	11.5	11.5	11.2	0.605	1.988
7	[456 , 40]	18.9	18.8	18.9	0.395	0.218
8	[456 , 80]	19.6	18.8	19.4	2.234	0.647
9	[456 , 120]	9.22	9.12	8.92	1.107	4.007

Table 3.4: The table demonstrates the need for normalization of the difference vectors in 3D, during implicit learning of normals. A normal requires at least two difference vectors in 3D to fully define it, and ideally require them to be of similar magnitude in order to eliminate any bias towards a particular direction. We here compute (for 9 points in the image) the magnitude of the maximum penalty for normal inconsistency in horizontal and vertical directions. The penalty is maximum when the normal is parallel to either of the difference vectors respectively. Since magnitude of the unit normal is 1, and the cosine angle is also 1 in the parallel case, the penalty magnitude is equal to the norm of the respective difference vector itself. We can see from the table that there is indeed an imbalance in the magnitude of penalty in the two directions. Note that the dense depth map shown here that is used to compute the penalties is generated by in-painting the corresponding sparse LIDAR map using our approach in Chapter 5. The normal map is generated from the dense depth map for visualization purposes only.



Stage	1	2	3	4	5	6	7	8	9	10
Layer	Conv.	Conv.	Conv.	Conv.	Conv.	Conv.	Conv.	Conv.	Conv.	Full
Input res.	561x427	71x55	36x28	36x28	36x28	36x28	36x28	36x28	36x28	36x28
In. channels	3	64	256	256	256	256	832	512	128	16
kernel size	11	5	3	3	3	3	3	1	1	
Output res.	141x108	71x55	36x28	36x28	36x28	36x28	36x28	36x28	36x28	36x28
Out. channels	64	256	256	256	256	512	512	128	16	3
Activation	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU		
Normalization	LRN	LRN								L2
Pooling (k. size)	Max(3)	Max(3)			Max(3)					

Figure 3.10: Our shallow CNN architecture for fast but less accurate surface normal prediction. There are nine convolutional layers (Conv.) and one fully connected layer (Full); In stages 1 to 8 we use Rectified Linear Units (ReLU) as activation, and Local Response Normalization (LRN) in stages 1 and 2. We apply L2 normalization to enforce a unit normal vector constraint in the output.

predicted surface normals?, *are lower-level information, such as shading, vanishing points and lines implicitly learned and utilized?*, and *which of these information contributes most to the accuracy of surface normals?* The same analysis can be carried out on CNN depth predictors.

Progressive downsampling of feature maps is a useful design choice to increase the input receptive field for each of the network's output without the need for a very deep network. While this increases the network's efficiency, it comes at a price of loss in precision, i.e. when the output is a dense pixelwise prediction. Architectures such as U-Net (Ronneberger et al. [2015]) minimize this problem by concatenating or summing up feature maps of previous layers with upsampled versions of later layers.

During the early stages of this work we tried out a VGG-F based CNN model (Chatfield et al. [2014]) for normal prediction. Although this model is less accurate and produce coarse output, it runs twice as fast as Eigen and Fergus [2015]. Such a network may therefore be suited for the computation and memory budget of low-power mobile devices. In the proceeding text we describe this faster model and compare its output against that of state-of-the art normal predictors like Eigen and Fergus [2015] in order to visualize the trade-off.

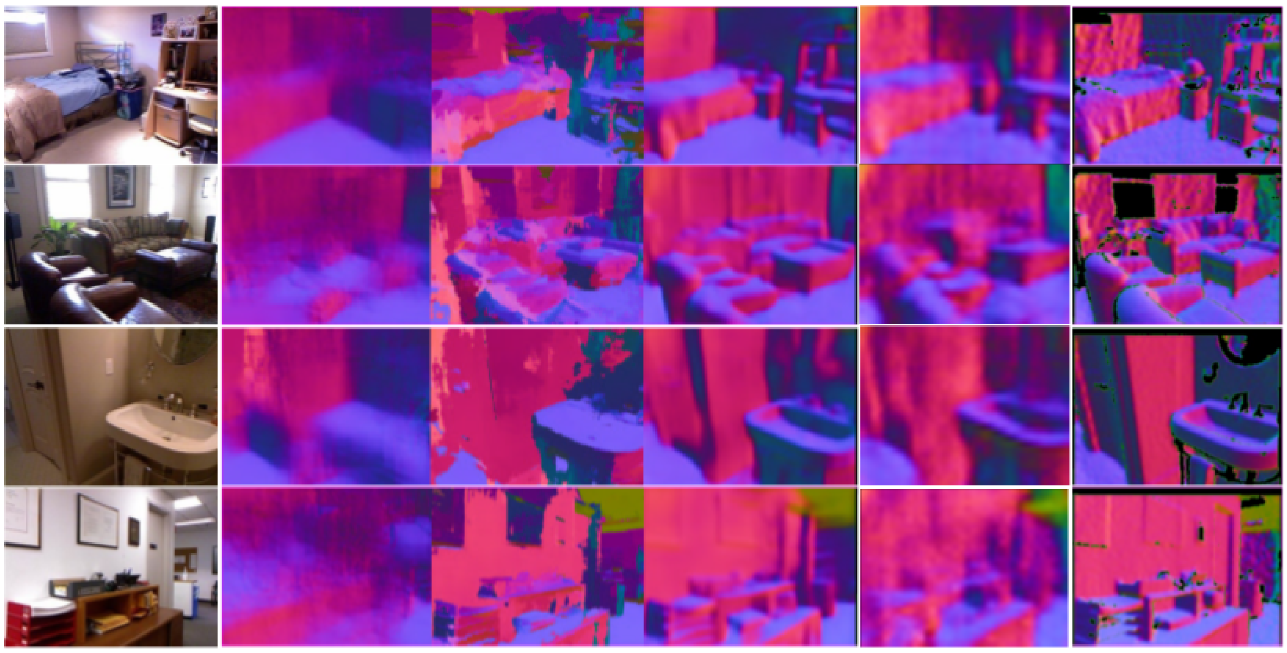


Figure 3.11: Comparison of different surface normal prediction methods against our proposed fast architecture. From left to right are: (1) the input image, (2) normals from Fouhey et al. [2013], (3) normals from Ladický et al. [2014], (4) normals from Eigen and Fergus [2015] that are used for all our experiments, (5) normals from our fast but less accurate model proposed in this section, and (6) groundtruth normals computed from Kinect depth data.

Our shallower model was trained on the raw NYU-Depth V2 dataset (Silberman et al. [2012]) containing 590 scenes in total, with 335 scenes used for training based on their train-test scene split. Given depth maps from the Kinect sensor for each frame in the scene, the ground truth surface normals were generated by fitting local least square planes on the world points corresponding to pixel coordinates in the image using the toolbox provided by Silberman et al. [2012]. In order to introduce additional variation in the training data, we augment the train set with the SUN-RGBD dataset (Song et al. [2015]) which in general consist of more accurate depth maps than the ones in the NYU raw dataset. We split the SUN RGB-D dataset into test and train images given their test-train split, however since this train split contains test images from the NYU dataset we remove these and add only the train images as specified in the NYU dataset split. The final training set consists of 292,224 RGB images and corresponding depth maps. We randomly shuffle the data prior to training (since each frame in a particular scene contains a lot of overlap) in order to increase stochasticity when running batch-based gradient descent (the stochasticity minimises the chance of the model weights getting trapped in a local optimum, however too much stochasticity may result in gradients that are too noisy). In order to enforce the network to predict normalised surface normals we add a L2 normalisation layer before the loss function. Our loss is the Euclidean distance between the normalised surface normal output and groundtruth unit normals. Alternatively one may use the negative dot product between the two normal vectors as the loss as in Eigen and Fergus [2015].

Since the groundtruth surface normal maps have missing values (resulting from the groundtruth depth maps also having missing values), we handle this problem during training by enforcing that the loss should be zero when groundtruth data is absent. We train our CNN model in four stages: first

	Fouhey et al. [2013]	Ladický et al. [2014]	Eigen and Fergus [2015] (VGG-16)	Ours (VGG-F)
Angle Distance Mean (deg)	37.7	35.5	22.2	31.3
Runtime	-	-	40ms	18ms

Table 3.5: Surface normals prediction accuracy measured in degrees against the ground truth where available. Groundtruth normals were computed from Kinect depth data using the toolbox in Silberman et al. [2012]. Note that it is difficult to compare runtimes for the first two methods due to different hardware/software configuration, however they are likely to be much higher than that of Eigen and Fergus [2015].

stage involves replacing the final fully connected layer with a convolutional layer, and training the model weights. In the second phase we replace the final convolutional layer with a fully connected one while keeping the weights of all the previous convolutional layers fixed. This enables fast training of the fully connected layer weights. We then finetune the weights of the overall model. Note that in the three phases of training, we fix the weights of the first 5 convolutional layers (initialized from a VGG-F model that is pre-trained on ImageNet data). In the fourth stage we finetune the entire model. This finetuning however only results in slight improvement in accuracy compared to the other three phases.

Figure 3.10 summarises our CNN architecture. Note that we concatenate the outputs of pooling layer 1 (after downsampling) and pooling layer 2 with convolutional layer 6 (across channels), and then feed the concatenated input into convolutional layer 7. This was done in order to enforce a mix of low-level and high-level features and allow the network to learn a better feature representation for surface normals more easily. We implement our model on top of the open-source Caffe C++ framework (Jia et al. [2014]) which provides a fast solution for both training and evaluating our CNN model.

Figure 3.11 shows a comparison of different surface normal prediction methods against our proposed fast architecture, and Table 3.5 provides a quantitative evaluation of the same. We can see that Eigen and Fergus [2015] produces more accurate and detailed results but take twice the runtime. The results reaffirms two things: (1) a deeper network is helpful to capture more complex relationships between image patches and surface normals, and (2) neural network-based approaches, even with our shallow architecture, outperform previous machine learning-based normal estimation methods (Fouhey et al. [2013]; Ladický et al. [2014]).

Apart from being a shallow network, a major contributor to our low accuracy is the low resolution output of our network. Future work could experiment with a better upsampling strategy. Interesting directions in improving neural network efficiency are proposed in the works of MobileNets (Howard et al. [2017]), SqueezeNet (Iandola et al. [2016]), Deep Compression (Han et al. [2015]), and Binarized Neural Networks (Courbariaux et al. [2016]), among several other recent work.

3.6 Conclusion

In this work we presented a simple yet efficient solution that jointly exploits low-level geometry-based photometric evidence and high-level scene information captured from a multi-scale CNN architecture in the form of surface normals, for improving the accuracy of dense reconstructions in cases where otherwise there is very little photometric evidence. It was seen that incorporating learnt surface orientations enabled smooth and accurate reconstructions especially in areas with little photometric evidence to guide the solution. Deep learning has enabled prediction of geometry of objects and scenes directly from a single image and this alleviates the need for prior assumptions about scene structure, and handcrafted scene priors that are otherwise required for dense reconstruction. It was also seen that these networks are capable of generalizing to new types of environments well enough for practical use. We believe this work is a step forward in unifying the two complementary tasks of 3D reconstruction and scene understanding, aiding purely vision-based autonomous robots.

Bibliography

A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5965–5974, 2016.

S. Y. Bao and S. Savarese. Semantic Structure from Motion: A Novel Framework for Joint Object Recognition and 3D Reconstruction. In *Proceedings of the 15th International Conference on Theoretical Foundations of Computer Vision: Outdoor and Large-scale Real-world Scene Analysis*, pages 376–397, Berlin, Heidelberg, 2012.

S. Y. Bao, M. Chandraker, Y. Lin, and S. Savarese. Dense Object Reconstruction with Semantic Priors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1264–1271, jun 2013.

M. J. Brooks and B. K. Horn. Shape and source from shading. 1985.

A. Chambolle and T. Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2010.

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *British Machine Vision Conference*, 2014.

J. Civera, A. J. Davison, and J. M. Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

A. Concha and J. Civera. Using superpixels in monocular SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 365–372, may 2014.

A. Concha and J. Civera. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5686–5693, sep 2015.

A. Concha, M. W. Hussain, L. Montano, and J. Civera. Manhattan and Piecewise-Planar Constraints for Dense Monocular Mapping. In *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, 2014.

M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid. Dense Reconstruction Using 3D Object Shape Priors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1288–1295, jun 2013.

J.-D. Durou, Y. Quéau, and J.-F. Aujol. Normal Integration – Part I: A Survey. jun 2016. URL `https://hal.archives-ouvertes.fr/hal-01334349`.

D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

A. Flint, D. Murray, and I. Reid. Manhattan scene understanding using monocular, stereo, and 3D features. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2228–2235, nov 2011.

D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3392–3399. IEEE, 2013.

R. Framkot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):439–451, 1988.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *ArXiv e-prints*, Jun 2015.

R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

C. Godard, O. Mac Aodha, and G. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611. IEEE, 2017.

S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, May 2014.

C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys. Joint 3D Scene Reconstruction and Class Segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 97–104, jun 2013.

C. Hane, N. Savinov, and M. Pollefeys. Class Specific 3D Object Shape Priors Using Surface Normals. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014.

C. Hane, L. Ladicky, and M. Pollefeys. Direction matters: Depth estimation with a surface normal classifier. In *CVPR*, pages 381–389, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

D. Herrera C., J. Kannala, L. Ladický, and J. Heikkila. Depth Map Inpainting under a Second-Order Smoothness Prior. In *Image Analysis*, volume 7944, pages 555–566. Springer Berlin Heidelberg, 2013.

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590, 2017.

G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

K. Kolev, T. Pock, and D. Cremers. Anisotropic Minimal Surfaces Integrating Photoconsistency and Normal Information for Multiview Stereo. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ECCV'10, pages 538–551, 2010.

P. Kovesi. Shapelets correlated with surface normals produce surfaces. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 994–1001. IEEE, 2005.

A. Kundu, Y. Li, F. Dellaert, F. Li, and J. Rehg. Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In *Computer Vision – ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pages 703–718. 2014.

L. Ladický, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. Torr. Joint Optimization for Object Class Segmentation and Dense Stereo Reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012.

L. Ladický, B. Zeisl, and M. Pollefeys. Discriminatively Trained Dense Surface Normal Estimation. In *ECCV*, volume 8693, pages 468–484. 2014.

A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23:689–694, 2004.

G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1925–1934, 2017.

F. Liu, C. Shen, and G. Lin. Deep Convolutional Neural Fields for Depth Estimation from a Single Image. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015a. URL http://arxiv.org/abs/1411.6387.

F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *Technical report, University of Adelaide*, 2015b. URL http://arxiv.org/abs/1502.07411.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. In *ACM transactions on graphics (TOG)*, volume 23, pages 664–672. ACM, 2004.

M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2609–2616. IEEE, 2014.

V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. MonoFusion: Real-time 3D Reconstruction of Small Scenes with a Single Web Camera. In *ISMAR*, 2013.

V. Prisacariu, O. Kahler, M. Cheng, C. Ren, J. Valentin, P. Torr, I. Reid, and D. Murray. A Framework for the Volumetric Integration of Depth Images. *ArXiv e-prints*, 2014.

R. T. Rockafellar. *Convex Analysis*. Princeton landmarks in mathematics and physics. Princeton University Press, 1997.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. June 2015.

F. Steinbrücker, T. Pock, and D. Cremers. Large displacement optical flow computation without warping. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1609–1614. IEEE, 2009.

J. Stühmer, S. Gumhold, and D. Cremers. Real-time Dense Geometry from a Handheld Camera. In *Proceedings of the 32Nd DAGM Conference on Pattern Recognition*, pages 11–20, Berlin, Heidelberg, 2010. Springer-Verlag.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012.

K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6565–6574. IEEE, 2017.

Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.

R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analaysis and Machine Intelligence*, 21(8):690–706, 1999.

Chapter 4

Learning Deeply Supervised Good Features to Match for Dense Monocular Reconstruction

In this chapter we exploit the end-to-end feature learning capabilities of a CNN for directly addressing the multi-view correspondence problem. We propose a novel Convolutional Neural Network (CNN) architecture along with a feature learning scheme for fast pixel-wise regression of large receptive field features that are "good for matching". By replacing photometric error with our learned features-based error in our energy formulation for keyframe reconstruction in Chapter 3, we analyze the combined benefits of having both a learned normal prior and learned features for matching in the context of real-time dense monocular reconstruction. Finally, we also demonstrate the usefulness of our learned features for the task of unsupervised learning for single view depth estimation in the KITTI dataset. The chapter concludes with a discussion of several avenues for future work.

4.1 Introduction

Many visual SLAM, SfM, and 3D reconstruction frameworks have been proposed over the years, yet most of these methods at their core rely upon establishing correspondences between multi-view images, either explicitly or implicitly, and use this information to solve for the locations of 3D landmarks and/or camera pose. While matching using handcrafted features such as ORB (Oriented FAST and rotated BRIEF) is mostly sufficient in cases where the desired output is the trajectory of a moving robot (Mur-Artal et al. [2015]), generating accurate dense correspondences between images becomes more crucial in the case of dense mapping (as well as direct dense tracking). Reliance on hand-crafted features or raw RGB pixel values is the major bottleneck of dense SLAM as matching ambiguities arise due lack of unique local texture, repetitive texture in the image, appearance distortion due to change in perspective, change in lightning, motion blur and occlusions. Hence, as we saw in the previous chapter, vision-based multi-view dense reconstruction systems greatly rely upon scene priors to compensate for the unreliability in dense pixel matching.

Recently it has been shown that CNNs, in addition to solving various high and mid-level tasks

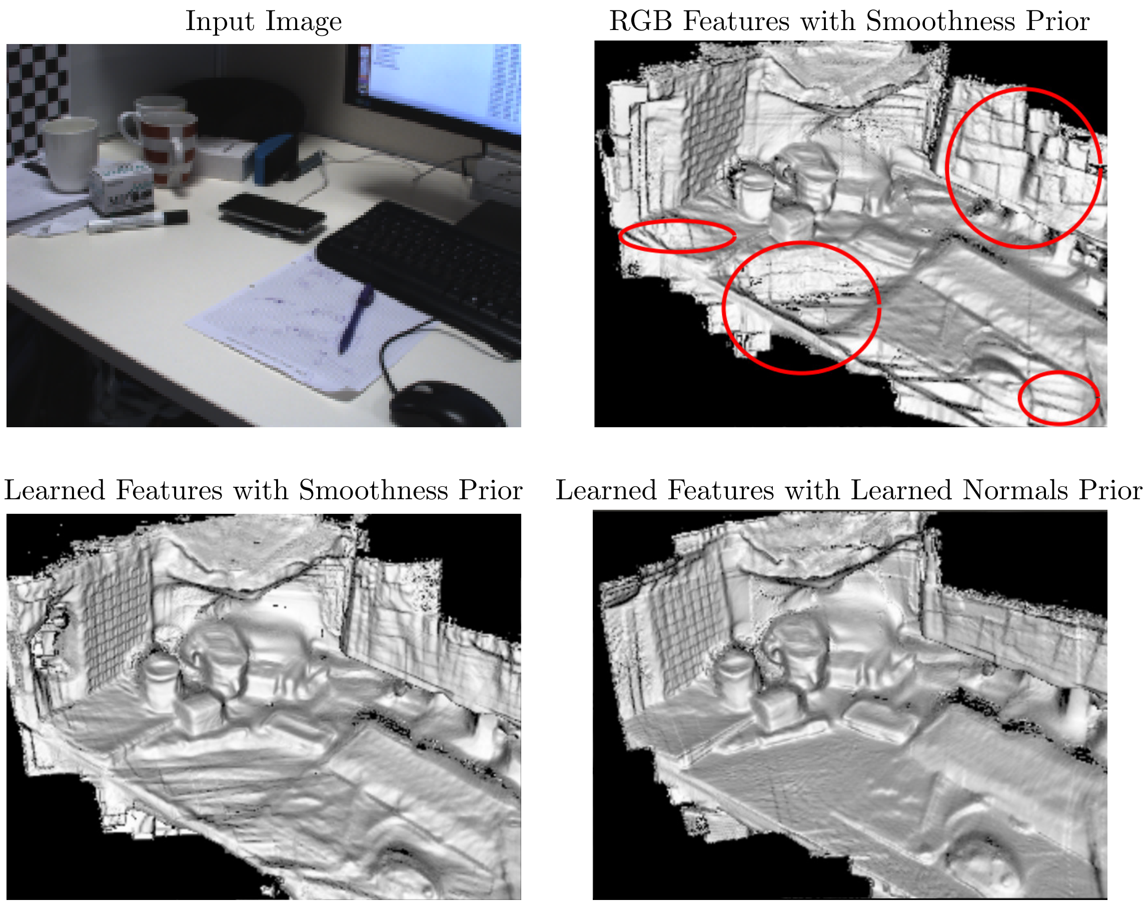

Input Image
RGB Features with Smoothness Prior
Learned Features with Smoothness Prior
Learned Features with Learned Normals Prior

Figure 4.1: Replacing color features with learned deep features and hand crafted priors with learned priors improves the reconstruction quality. Notice the dent in table, the very structured error in fusing the depths of the monitor and wrongly reconstructed notebook in the reconstruction which is obtained by using a RGB cost volume with smoothness regularization of DTAM (Newcombe et al. [2011b]). Most of these errors can be fixed by learning deep-features to generate the cost-volume and using learned priors to replace the smoothness prior.

(Eigen and Fergus [2015]; Lin et al. [2017]), are also capable of accurately solving low-level vision problems like predicting dense optical flow/disparity (Ranjan and Black [2016], Kendall et al. [2017b]) or depth and motion from two-views (Ummenhofer et al. [2016]) outperforming traditional vision methods which rely on handcrafted features and loss functions. However they have mainly been used for solving single or two-frame problems. It is not straightforward to extend standard end-to-end learning frameworks to use data from arbitrary number of views without increasing memory and computational requirements while training and testing.

In a typical visual SLAM system where the camera captures images at 30Hz, it is always beneficial to make use of all the information available. In this work, we take a different approach of learning dense features *good for matching* which are *fast* to compute and can directly be used in the existing dense monocular reconstruction systems for reliable correspondence estimation.

More specifically we want to automatically learn features via a CNN to deal with the ambiguities that occur when matching along epipolar lines for per-pixel depth estimation, especially when presented with images captured with a handheld monocular camera. To this end we introduce a novel CNN-based deeply supervised training scheme for feature descriptor learning, and a fully convolutional multi-scale network design that can help efficiently capture both local and global pixelwise context, and simultaneously regress for a high dimensional feature vector for each pixel in the image. In contrast to previous feature learning approaches, we propose to construct and minimize a multi-view cost volume loss during training where the predicted feature for *each* pixel in the reference image is compared with those for a range of pixels along the corresponding epipolar line in the live frame (one of which will be the right match).

The minimization is done with respect to predicted features over the *entire* cost volume during training and thus millions of training instances are incorporated, and into a single training iteration, allowing for efficient training with more stable (low variance) gradient-based neural net weight updates. The efficient computation of our loss is enabled by the large parallel compute capability of GPUs. We apply our cost volume loss after every feature map downsampling stage in order to regularize the network to learn good features to match at every image scale. Moreover since our training loss is based on the same cost volume loss minimised at test-time for estimating depth, our belief is that it will help the network learn the optimal features suited for dense correspondence in a monocular reconstruction setting given the training data and model capacity. Similar to Schmidt et al. [2017] our training framework is self-supervised, requiring only raw RGB-D data (e.g. as captured by Kinect) and the camera trajectory that can be estimated from the data itself using a method like RGB-D ORB-SLAM (Mur-Artal and Tardós [2016]) or Kinect Fusion (Newcombe et al. [2011a]) for rigid scenes.

We also integrate our feature descriptors into the real-time dense monocular reconstruction system in Chapter 3 with the learned normal prior. We show that the combined system leads to further improvement in dense monocular reconstructions in challenging indoor environments (Figure 4.1). To the best of our knowledge this combined system is the first framework for real-time monocular dense reconstruction to harness the learning capabilities of CNNs in terms of *both* geometric scene understanding (in the form of learned surface normals) and dense correspondence (in the form of "good features to match") given arbitrary number of input images.

4.2 Background and Related Work

SLAM Methods with Hand Crafted Features: Most popular SLAM methods like PTAM (Klein and Murray [2007]) and ORB-SLAM (Mur-Artal et al. [2015]) use handcrafted features successfully for sparse matching and solving for a sparse map and camera trajectory. LSD-SLAM (Engel et al. [2014]) uses direct RGB alignment for solving for both pose and map, and is able to generate a semi-dense map aided by depth regularization. DTAM (Newcombe et al. [2011b]) is also based on direct RGB alignment but is able to generate fully dense depth maps by accumulating multiple frames for keyframe reconstruction, and utilizing a smoothness prior to help resolve matching ambiguities. While recent work has begun exploring the use of learned priors for reconstruction (Fácil et al. [2016]; Tateno et al. [2017]; Weerasekera et al. [2017]), these methods still rely on low-level color/feature matching for dense correspondence.

End-to-end CNNs for Stereo: Ummenhofer et al. [2016] propose to train a network for depth, camera motion, and optical flow prediction from two frames for indoor scenes, and Kendall et al.

[2017b] propose to train a deep network end-to-end for disparity prediction outdoors, to give state-of-the-art results for stereo reconstruction in NYU and KITTI datasets respectively. These two-frame reconstruction architectures however are not very easily extensible to use more than two frames — a limitation we address in this work. In particular, our work is inspired by Kendall et al. [2017b], which constructs a cost-volume with learned features for two frames by simply concatenating the features of these images for all possible candidate matches. This stereo cost-volume is further passed to a deep 3D convolutional network which implicitly acts as a regularizer to finally select the correct matches out of these candidates. Instead of learning this regularization network, in this work we propose to make the matching decision on per-pixel basis directly based on the cost-volume by explicitly forcing the correctly matched features to have small distances (and large otherwise). At test time we can then infer the depth of each pixel in the reference image *based on the same/similar distance metric in the learned feature space summed over multiple live frames*, which end-to-end stereo networks listed above do not allow.

Feature Learning with CNNs for Low-Level Vision: Feature learning methods designed for good matching e.g. MC-CNN (Zbontar and LeCun [2016]) or LIFT (Yi et al. [2016]) are often patch based Siamese like architectures. Unlike these methods, our architecture is fully convolutional allowing for efficient dense feature extraction during training and testing and with flexibility to match each pixel relying potentially on a receptive field of as large as the entire image. Related to our work is recently published self-supervised visual descriptor learning method of Schmidt et al. [2017] for matching. At test time, the learned features are used for model to frame alignment under extreme scene variations. Another related work is Universal Correspondence Network (Choy et al. [2016]) that learn a network for geometric and semantic feature matching. In both the latter methods, each pixel in one image (the keyframe image) is compared with positive and negative matches in another image (the live image) and the network is forced to learn features $\mathbf{f}_{ref}$ and $\mathbf{f}_{live}$ (sharing the same network weights) that minimize the following contrastive loss:

$$\frac{1}{2N}\sum_{i}^{N} s_i||\mathbf{f}_{ref}(\mathbf{x}_i) - \mathbf{f}_{live}(\mathbf{x}_i^{'})||^2 + (1 - s_i)\text{max}(0, m - ||\mathbf{f}_{ref}(\mathbf{x}_i) - \mathbf{f}_{live}(\mathbf{x}_i^{'})||)^2 \tag{4.1}$$

The above correspondence contrastive loss consists of two terms. The first term minimizes the distance between positive pairs (when $s_i = 1$) and the second term pushes negative pairs to be at least margin m away from each other (when $s_i = 0$). This is summed over N training examples. The cross-entropy loss that we employ (as described in Section 4.3.2) enforces a similar effect by forcing the *probability* of the true match to be 1 only at the corresponding location in the other image, and 0 elsewhere. Both of these losses are employed commonly in the literature for image classification and feature learning tasks and further empirical evidence is needed to justify the choice of one loss over the other. However, the cross-entropy loss has the benefit of not needing to hard-code a distance m in feature space for negative matches as in the contrastive loss.

Some other main differences exist between our feature learning method and previous work: (1) we use deep supervision (we minimize the feature matching error at different scales in the network to learn multi-scale features that are good to match), (2) we design and use a fast and efficient neural net architecture specially suited for dense correspondence (which is independent of spatial transformer layers as used in Choy et al. [2016] and Yi et al. [2016] to add to efficiency), and (3) previous work on feature learning have relied upon randomly selected negative samples (e.g. nearest neighbor sampling around the true match at *integer* pixel locations in Choy et al. [2016]) — in our case, by sampling uniformly along the epipolar line we generate positive and negative training examples at *sub-pixel* level (through bilinear interpolation in feature space) and this is more reflective of the type of matching during dense reconstruction. Moreover, existing methods have not been extensively analyzed or integrated into a dense reconstruction framework.

4.3 Method

In this section we describe our network architecture and training and inference time setup, and discuss the motivations behind the design choices.

4.3.1 Neural Network Architecture

Our network consists of five blocks B1-B5 (Figure 4.2), each block consisting of 3 convolutional layers. All convolutional layers have a kernel size of 3x3 and 32 output filters, which we found to be a good middle ground for computational speed and matching accuracy. While we have tried other variations of our network with convolutional kernel sizes 9x9, 7x7, 5x5, interestingly the one with convolutional kernel size 3x3 in each block enabled us to achieve the highest matching precision. We believe that this is because explicitly limiting the receptive field, especially at the earlier layers, helps the network learn features that are invariant to appearance distortion.

The first two layers of each block consist of ReLU activation units for adding non-linearity to the network. The first layer of each block has a stride of 2 for 2x downsampling after each subsequent block. Downsampling in Block B1 is optional but adds to the efficiency, at a slight trade-off of precision. The progressive down sampling rapidly increases the receptive field for a pixel without the need for a deeper network and helps the network capture contextual information which is key for successful matching. Not having a very deep network also prevents the network from overfitting to a particular dataset, and helps it learn more generic descriptors that are useful for matching even in completely different types of environments (Figure 4.4). In our network, the maximum potential receptive field for a pixel is roughly half the input image resolution which is significantly larger than in work like Zbontar and LeCun [2016] which limit the receptive field to 9x9 patches.

The downsampling however comes at a cost of reduced precision in the matching. This is because coarse features, although useful for matching large textureless regions, are too abstract for fine-grain pixel level matching which is required for precise depth estimation. Inspired by the U-Net architecture (Ronneberger et al. [2015]) we upsample each coarse prediction and sum it with the preceding fine predictions (as shown in Figure 4.2) to produce our final feature output. Doing so explicitly embeds local information about a pixel into its contextual representation enabling precise sub-pixel level matching.

The final output of our network is therefore a 32 dimensional feature vector for each pixel in the image, encoding both contextual and low-level image information. Each upsampling operation on the output of each coarse block is done through a single learnable deconvolution layer with a kernel size

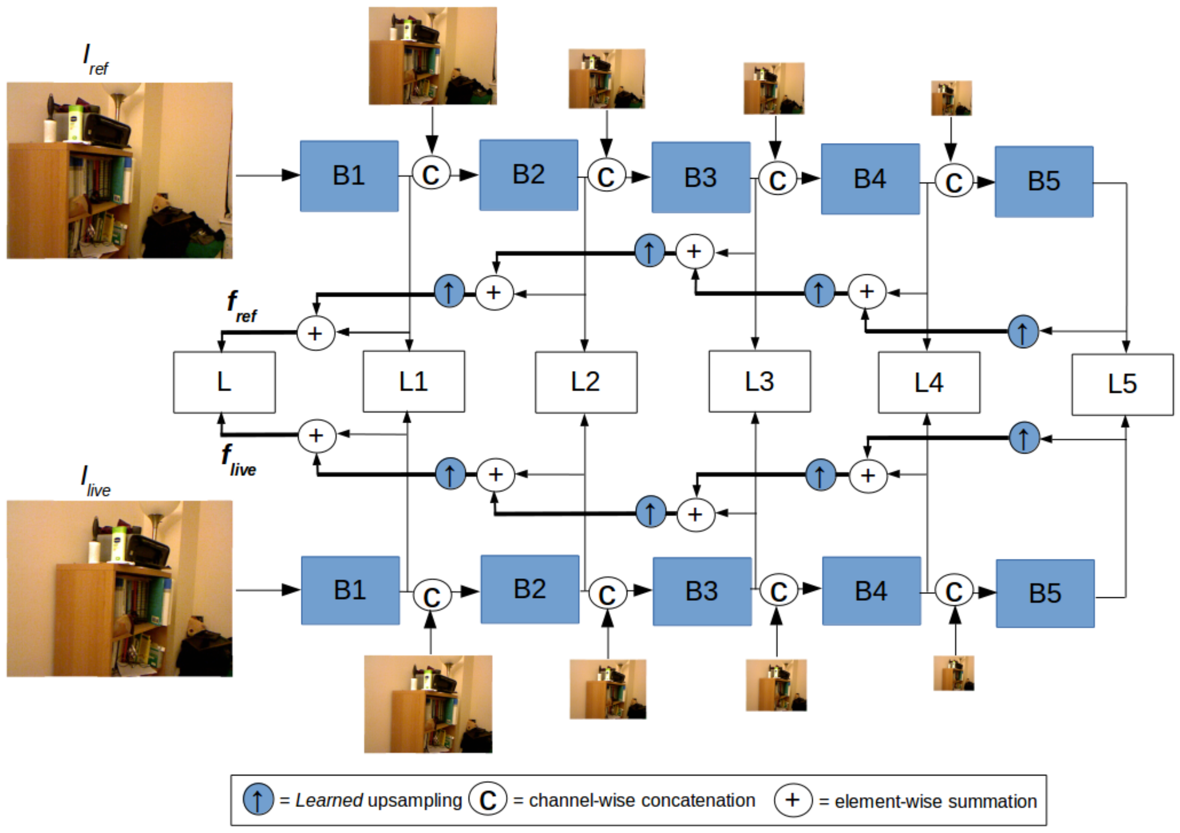


Figure 4.2: Overview of the proposed multi-scale network architecture and training setup. Blocks B1-B5 each consist of 3 convolutional layers (kernel size=3, pad = 1, output filters=32) with the first layer in each block (except Block B1) having a stride=2, and the rest having stride=1, so that the features are downsampled by a factor of two after each block. The first two layers in each block have ReLU activation units. The upsampled coarse features are summed up with the fine features to produce a feature comprising of contextual information as well as fine image details. At train-time both the reference and live images (I_{ref} and I_{live}) are passed into the network as a batch in two independent streams that share weights. The final 32 dimensional output features ($\mathbf{f_{ref}}$ and $\mathbf{f_{live}}$) for the reference and live image respectively as well as their intermediate output features are used as input to the cost volume loss layers L,L1-L5 for deep feature supervision at every image scale. Note that bold lines do not intersect with thin lines. The forward time for the network for a single image to produce a dense feature map is only $\approx 8ms$ on a Nvidia GTX 980 GPU.

of 5x5 and stride of 2. Making this layer learnable results in improved results over a simple bilinear upsampling, showing that in this case it is better to allow the network to learn a more appropriate upsampling method. Furthermore, concatenating a bilinearly downsampled version of the input image with the output features from each block before passing them as input into the subsequent coarser blocks also improves the matching result. This shows that there is still information present in the low-level features that can complement the abstract ones in order to benefit the pixelwise matching at lower resolutions. All input images are mean (of training set) subtracted but not normalized as this gave the best performance in practice.

Although the multi-scale features from our network can be used independently for coarse-to-fine matching at test time, we perform matching using just the final *aggregated* high resolution feature output of our network instead as the contextual information contained in it allows for a large basin of convergence and makes the solution less sensitive to initialization (see Section 4.4.1). Our network is fast to evaluate ($\approx 8ms$ on a Nvidia GTX 980 GPU) making it suitable for use as a feature extractor in real-time SLAM. Another advantage of our architecture is that it provides the flexibility for one to discard the coarser blocks if needed, as the features that are generated by each block are *explicitly trained* to be suitable for matching through deep supervision (see Section 4.3.2). For example one can choose to remove Blocks B2-B5 and simply use features from Block B1 if further efficiency is desired without compromising too much on accuracy (refer Figure 4.4). One can also reduce the feature vector length from 32 to improve the matching speed, or learn a feature vector with higher dimensionality to improve the matching accuracy. In our CUDA run-time implementation we make use of $float4$ data structures and 3D texture memory for improving GPU memory access times.

4.3.2 Training

The output of our network are dense pixelwise features for a given image. Given a pair of images (that are 30 frames apart in our training setup), the goal is to learn suitable features that give a minimum matching cost *only* between a pixel $\mathbf{u}_p$ in the reference image and the corresponding matching pixel in the live image, that lies on the corresponding epipolar line and is defined by the 3D location of $\mathbf{u}_p$ and the relative camera motion. To do this, first we discretize the (inverse) depth space ρ_p into $K = 256$ bins ($\rho_p = \{\rho_p^1, ..., \rho_p^K\}$, with $\rho_p^1 = 0$ and $\rho_p^K = 4$), and convert this continuous matching problem into K class classification problem. Note that the number of inverse depth bins and the range of inverse depth labels need not necessarily be the same at test time. This is a significant advantage of our approach over an end-to-end approach like Kendall et al. [2017b] — in their case parameters like number of inverse depth bins and range of inverse depth labels are tied to the network architecture since their cost volume creation step is followed by a learnable 3D convolution-based depth regularization network that refine the matching probability at each inverse depth bin of the stereo cost volume.

We create a discrete multi-view cost-volume (as in DTAM (Newcombe et al. [2011b]) and in Chapter 3), but where the matching cost for each pixel in the reference image (I_{ref}), for each discrete inverse depth label is computed (in parallel on the GPU), based on the *squared learned feature distance* (as opposed to the more robust L1 distance as used at test time) between the pixel in the reference frame and the hypothesized matching pixel in the live frame (I_{live}) that lies on the epipolar line.

More specifically, our goal is to find the optimal features $\mathbf{f}_{ref} = f(\mathbf{w}, I_{ref})$ and $\mathbf{f}_{live} = f(\mathbf{w}, I_{live})$ that are a function f of the shared neural network weights $\mathbf{w}$ and I_{ref} and I_{live} respectively, that minimize the KL-divergence between the estimated probability $P_\phi(\rho_p^l)$ (derived from the cost volume E_ϕ as defined further below) and the true probability $P(\rho_p^l)$ which is equal to 1 when $\rho_p^l = \rho_p^*$ (the groundtruth inverse depth label), and is equal to 0 otherwise, and additionally minimise the KL-divergence between $1 - P_\phi(\rho_p^l)$ and $1 - P(\rho_p^l)$:

$$\sum_p \sum_l P(\rho_p^l)(lnP(\rho_p^l) - lnP_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})) + (1 - P(\rho_p^l))(ln(1 - P(\rho_p^l)) - ln(1 - P_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})) \tag{4.2}$$

which reduces to the cross-entropy loss:

$$\sum_p \sum_l P(\rho_p^l)(-lnP_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})) + (1 - P(\rho_p^l))(-ln(1 - P_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})) \tag{4.3}$$

where,

$$P_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live}) = \frac{e^{-E_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})}}{\sum_{k=1}^{K} e^{-E_\phi(\rho_p^k, \mathbf{f}_{ref}, \mathbf{f}_{live})}} = \sigma(-E_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})) \tag{4.4}$$

and

$$E_\phi(\rho_p, \mathbf{f}_{ref}, \mathbf{f}_{live}) = \left\| \mathbf{f}_{ref}(\mathbf{u}_p) - \mathbf{f}_{live}(\pi(T_{nr}\pi^{-1}(\mathbf{u}_p, \rho_p))) \right\|_2^2 \tag{4.5}$$

and $\sigma(.)$ is the soft-max operation.

$T_{nr} \in \mathbf{SE}(3)$ is a matrix describing the transformation of a point from camera coordinates of $\mathbf{I}_r$ to that of $\mathbf{I}_n$. $\pi(.)$ is the projection operation, and $\pi^{-1}(.,.)$ is the back-projection operation, such that $\pi^{-1}(\mathbf{u}_p, \rho_p) = K^{-1}\dot{\mathbf{u}}_p/\rho_p$, where K is the camera intrinsics matrix, and $\dot{\mathbf{u}}_p = (u, v, 1)^T$ is $\mathbf{u}_p$ (a pixel in the reference image) in homogeneous form.

Note that since the length of the epipolar line in the live image (for the specified inverse depth range) can vary depending on the camera baseline and we sample a fixed number of bins, we perform bilinear interpolation in feature space to obtain features corresponding to each inverse depth hypothesis at non-integer pixel locations. Furthermore, since the epipolar line can stretch out of the live image, we set a high cost m (e.g. $m = 10$) for the negative matches that fall outside. If, on the other hand, the correct match to a pixel in the reference image lie outside of the live image, no loss is applied at all for such pixels in the reference image.

In order to mitigate the effects of discretization and enforce a more refined match we further minimize the regression loss in the same manner as Kendall et al. [2017b], by performing a dot product between the discrete inverse depth labels and the estimated set of probabilities $P_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})$, to get a single $\hat{\rho}_p$ for each pixel, and then minimising the Euclidean distance of $\hat{\rho}_p$ to the groundtruth inverse depth ρ_p^*. On top of that we also minimize the Euclidean distance of the inverse of $\hat{\rho}_p$ ($\hat{d}_p$) to the inverse of ρ_p^* (d_p^*). In other words we want to make the expected inverse depth/depth value to be as close as possible to the groundtruth.

Our overall loss function could therefore be written as:

$$
\begin{aligned}
&\sum_p \sum_l P(\rho_p^l)(-ln P_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})) + (1 - P(\rho_p^l))(-ln(1 - P_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})) \\
&+ \sum_p \Big(\lambda_\rho (\hat{\rho}_p - \rho_p^*)^2 + \lambda_d (\hat{d}_p - d_p^*)^2\Big)
\end{aligned}
\tag{4.6}
$$

where,

$$
\hat{\rho}_p = P_\phi(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})\rho_p^l. \tag{4.7}
$$

We experimentally found that using a combination of the aforementioned classification loss and the two regression losses in both depth and inverse depth space, improved the precision of $\hat{\rho}_p$. We empirically set $\lambda_\rho = 5, \lambda_d = 1$ in order to balance the magnitude of the three types of losses, and to balance the bias in precision between depth and inverse depth label space. Automatically learning λ_ρ and λ_d to balance our different loss components similar to what is done in Kendall et al. [2017a] is an avenue for future work.

Since our network regresses multi-scale features, similar to Xie and Tu [2015], we explicitly perform deep supervision by minimizing both classification and regression losses with respect to the features at each scale independently (L1-L5), as well as the aggregated feature (L in figure 4.2). Note that the camera intrinsics required for cost volume generation are scaled to suit the output feature resolution at each block. We obtain groundtruth depth maps at each lower resolution scale through nearest neighbour sampling of the full resolution depth map. In comparison, similar works to ours like Choy et al. [2016]; Schmidt et al. [2017] have relied upon a single contrastive loss function at the end of the network. In addition to providing the flexibility of discarding blocks of the network in trade for computation speed, this deep supervision acts as a good regularizer for the network resulting in fast convergence during training. Our network is trained from scratch with Xavier initialization using the Caffe framework (Jia et al. [2014]).

4.3.3 Keyframe Reconstruction Using Deep Features

Once we have learned the multi-scale features as specified above, we can directly use them in mapping (and tracking) frameworks like that of DTAM (Newcombe et al. [2011b]) and Chapter 3. For mapping we can simply replace the RGB cost-volume with the learned deep-features based cost-volume. Our inference loss for keyframe reconstruction is defined as:

$$
E(\boldsymbol{\rho}) = \sum_{p \in \mathcal{P}} \frac{1}{\lambda} E_\phi(\rho_p) + E_{reg}(\rho_p), \tag{4.8}
$$

where E_ϕ is the dataterm as follows,

$$
E_\phi(\rho_p) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{f}_{ref}(\mathbf{u}_p) - \mathbf{f}_n(\pi(T_{nr}\pi^{-1}(\mathbf{u}_p, \rho_p))) \right\|_1 \tag{4.9}
$$

which computes the descriptor matching error for a keyframe $\mathbf{I}_{ref}$ accumulated over N overlapping frames. Note that the L1 norm is used here for more robust depth inference, while using the squared

L2 norm during learning of features led to lower depth/inverse depth regression losses.

E_{reg} consists of the prior terms. We experiment with both the inverse depth smoothness prior and our learned normal prior proposed in Chapter 3. λ in each case controls the regularization strength.

4.4 Experimental Results

4.4.1 Evaluation of Learned Features for Matching

We extensively evaluate the matching performance and desirable properties of our learned deep features against several baselines on a large subset of the NYU-D v2 dataset (Silberman et al. [2012]) and follow that up to further explore the generalization capability of learned features quantitatively on TUM (Sturm et al. [2012]) and ICL-NUIM (Handa et al. [2014]) datasets and qualitatively on KITTI (Geiger et al. [2013]) dataset. Note that our features used in all experiments are trained on the raw NYU-D v2 dataset, excluding all test scenes. This is followed by an experiment which uses our learned features for unsupervised depth estimation on the KITTI dataset, where we show, among other things, that features for matching can also be learned unsupervised.

Quantitative and Qualitative Results on the NYU-D v2 Dataset: We follow the same experimental set-up and the train/test splits of the NYU-D v2 dataset as that of Chapter 3. Camera motion (in metric units) for all the NYU train and test sequences were precomputed using RGB-D ORB-SLAM (Mur-Artal and Tardós [2016]) to isolate the mapping process. The mapping framework of DTAM and that of Chapter 3 will serve as our main baselines for this work.

Small sub-sequences of 61 frames (30 past and 30 future frames) were used to reconstruct all the keyframes in all the experiments. We show the improvement in depth estimation after replacing RGB features with our learned features for cost volume generation in both the baselines mentioned above. As in Chapter 3, we evaluate the accuracy of the depth maps based on the standard evaluation criteria used in the NYU-D v2 dataset (Eigen and Fergus [2015]).

To find optimal hyper parameter λ, we grid-search in a sensible range for each of the approach individually. Figure 4.3 visualizes the sensitivity of all the reconstruction approaches to the choice of regularization strength λ. After scaling λ for each method to within a constant normalization factor, it can be seen that all algorithms degrade gracefully when we deviate from the optimal choice of λ while the learned features make this choice less critical.

In Table 4.1 we report the best reconstruction accuracies obtained on the NYU-D v2 dataset with and without using our learned features. Using our learned features (instead of using RGB features) to create the cost volume allows for a significant improvement in reconstruction performance in terms of all evaluation criteria. In particular the percentage of accurately reconstructed points (with accuracy threshold greater than 1.25) improves from 83.4% to 93.6% with inverse depth smoothness-based regularization. A similar improvement is observed when our learned normal prior is used instead of the smoothness prior.

Figures 4.5, 4.6, and 4.7 show a visual comparison of the reconstructions generated in real-time using the proposed deep features against those generated using RGB features, with either the learned normals or smoothness prior, in parallel with ORB-SLAM monocular visual tracking (Mur-Artal et al. [2015]). As in Chapter 3, we fuse the depth maps of the keyframes obtained using each of the methods

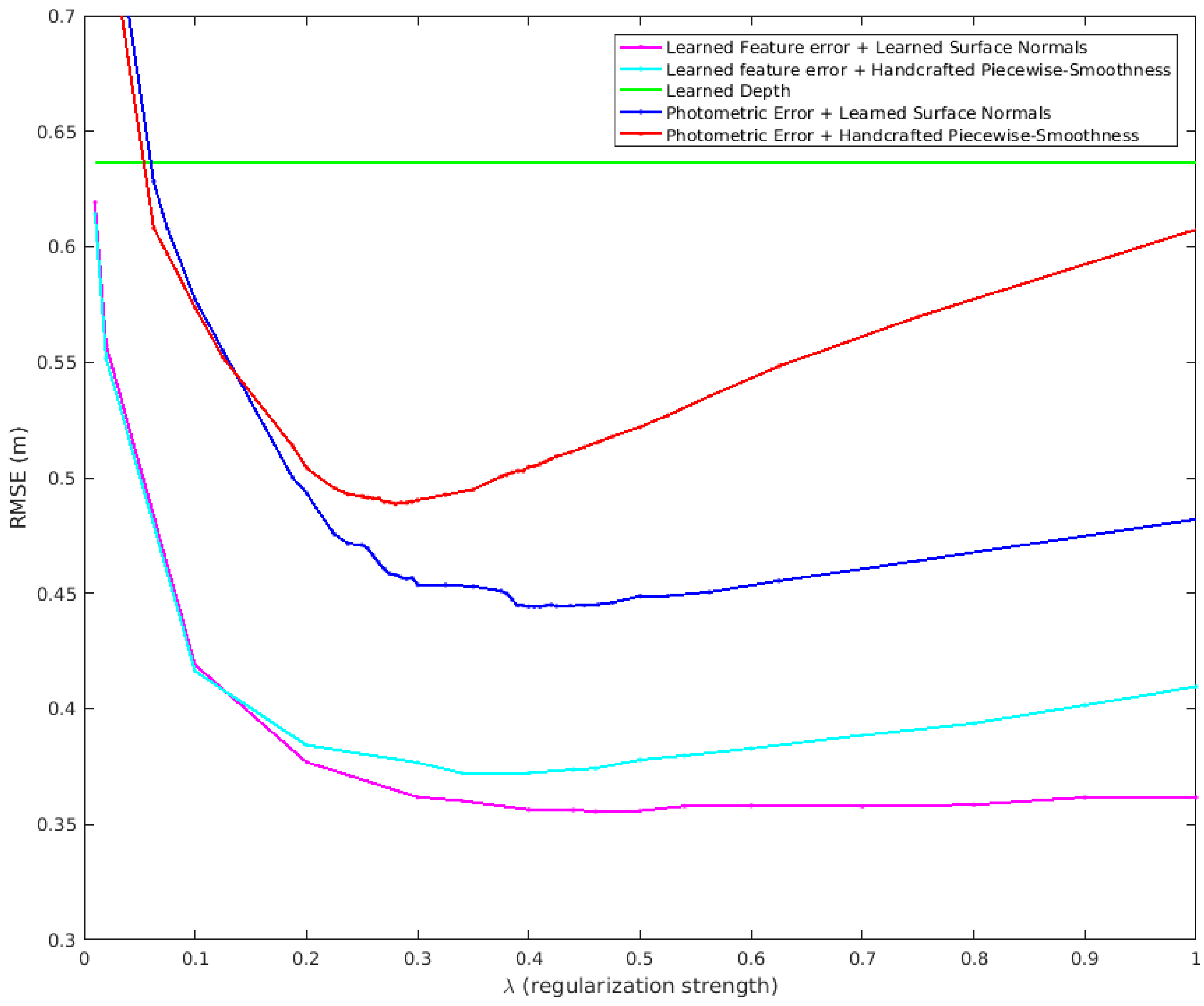


Figure 4.3: The effect of changing regularization strength on reconstruction accuracy. The reconstruction errors are significantly lower with learned features than with RGB features for matching. Using learned normal based prior of Chapter 3 with our learned features gives best results. Please note that as the cost-volume magnitude depends on the length and range of the used features, for the 2 plots corresponding to the learned feature error, the regularization strength λ is divided by a constant factor of 12.5 for better visualization alongside the 2 plots corresponding to RGB (photometric) feature error.

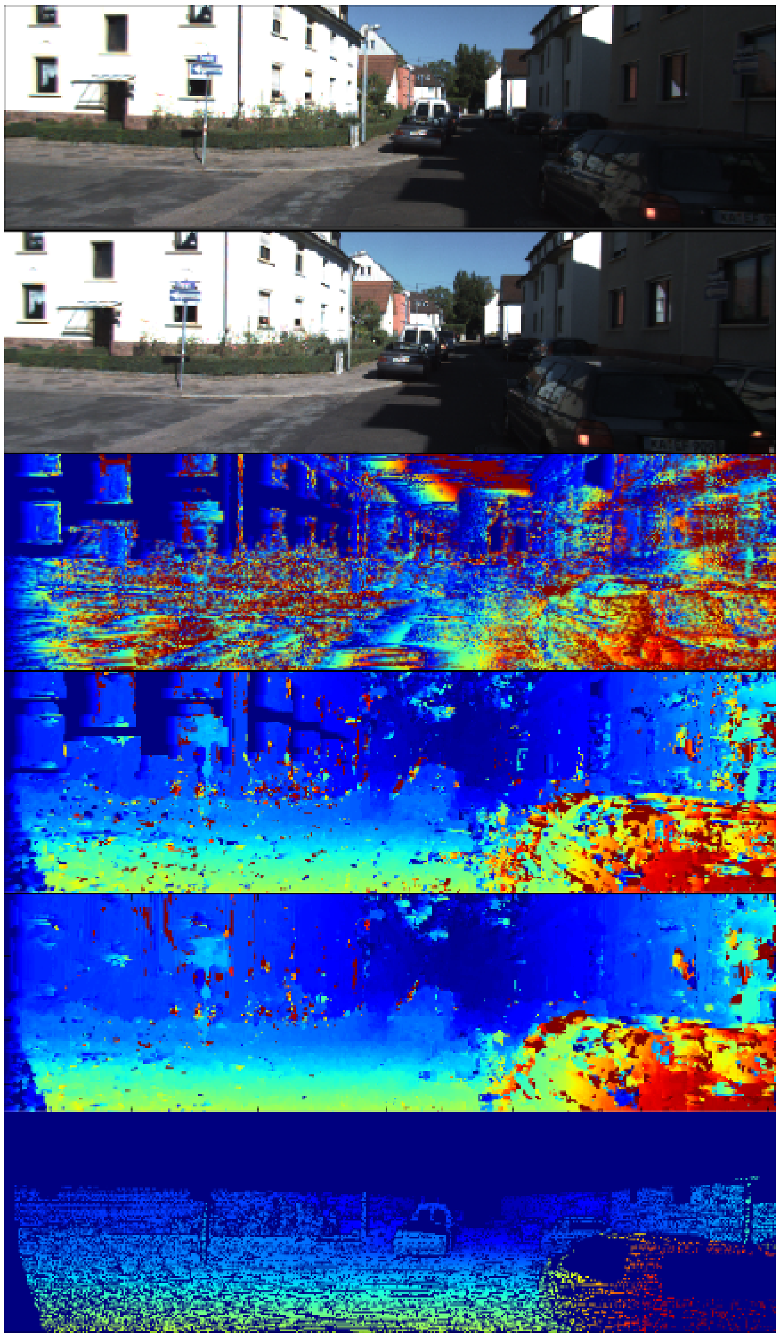


Figure 4.4: Generalizability of learned features trained on NYU dataset to a completely different type of scene on the KITTI stereo dataset without any finetuning. Note that no regularization was added on top, so this is just the pure matching result. From Top to Bottom: Reference Image, Live Image, Disparity map obtained by matching using rgb features, Disparity map obtained by matching using learned features from Block B1, Disparity map obtained by matching using all learned features combined, Groundtruth. Notice how explicitly combining coarse features with the fine features provides the necessary context to match large textureless regions while maintaining precision.

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CNN Depth	0.637	0.226	0.163	0.135	0.738	0.937	0.982
P.E. + Smoothness	0.522	0.206	0.123	0.111	0.834	0.949	0.979
P.E. + L. Normals	0.449	0.174	0.086	0.076	0.893	0.964	0.985
L.F.E. + Smoothness	0.372	0.143	0.067	0.054	0.936	0.978	0.990
L.F.E. + L. Normals	**0.356**	**0.135**	**0.058**	**0.049**	**0.948**	**0.981**	**0.991**

Table 4.1: Quantitative results on 25 raw NYU-D V2 Dataset test sequences. P.E. = Photometric Error. CNN Depth is the output of Eigen and Fergus [2015]. The average errors and accuracy are for keyframe reconstructions against Kinect depth maps (where valid depths are available). The results here are shown for the optimal lambda values for normals and smoothness regularizer.

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CNN Depth	1.141	0.368	0.227	0.261	0.543	0.820	0.923
P.E. + Smoothness	0.563	0.215	0.106	0.091	0.854	0.924	0.973
P.E. + Normals	0.558	0.215	0.103	0.089	0.863	0.922	0.969
L.F.E. + Smoothness	0.453	0.178	**0.079**	0.061	0.909	0.949	**0.981**
L.F.E. + L. Normals	**0.450**	**0.176**	**0.079**	**0.060**	**0.910**	**0.950**	**0.981**

Table 4.2: Quantitative results on the TUM dataset 'fr2_desk' sequence. P.E. = Photometric Error. CNN Depth is the output of Eigen and Fergus [2015].

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CNN Depth	0.829	0.426	0.295	0.261	0.472	0.781	0.905
P.E. + Smoothness	0.287	0.118	0.062	0.041	0.943	0.988	0.997
P.E. + Normals	0.214	0.094	0.047	**0.022**	0.963	0.993	**0.998**
L.F.E. + Smoothness	0.285	0.115	0.056	0.043	0.952	0.987	0.996
L.F.E. + L. Normals	**0.213**	**0.090**	**0.043**	**0.022**	**0.972**	**0.994**	**0.998**

Table 4.3: Quantitative results on ICL-NUIM dataset 'lr kt0' sequence. P.E. = Photometric Error. CNN Depth is the output of Eigen and Fergus [2015].

using InfiniTAM (Prisacariu et al. [2014]) as part of our framework for better visualization, and the results are shown and analysed in the figures. A significant advantage of the proposed deep features can be easily seen in these results visually. [1]

Generalization Capabilities Across Different Camera Models: Furthermore we quantify the importance of our learned context aware deep features over RGB matching on two more indoor datasets: TUM dataset 'fr2_desk' and ICL-NUIM synthetic dataset 'lr kt0', both of which contain images that possess different properties (e.g. camera intrinsics) from that of the NYU dataset. Tables 4.2 and 4.3 demonstrate that our deep features consistently improve the depth estimation accuracy in previously unseen datasets without need for fine-tuning.

Finally, we explore if our learned features are good for matching in extremely different outdoor road scenes where the camera model, objects and textures present in the scene, lighting conditions as well as camera motion vary from the indoor setup. The generalization capability of our network can be clearly observed in the example shown in Figure 4.4 where KITTI stereo images are matched using our features learned on the NYU-D v2 dataset. Our learned features show promising matching results, and also heavily outperform RGB matching in the two-frame matching case as expected.

Comparing Against Other Features for Dense Matching: We compare our learned features against some baselines which are also suited for the dense geometric matching task with their own unique strengths and weaknesses. Apart from raw RGB features as commonly used in dense SLAM systems like DTAM, we chose to use as baselines: 7x7 patches that are affine warped before matching (similar to what is used in DSO (Engel et al. [2017])), ResNet-50 Conv1 features (He et al. [2016]) pre-trained on ImageNet (with a receptive field of 7x7), DenseSIFT (used for dense optic flow in Liu et al. [2016]), and features from Block B1 of our network with a receptive field of 7x7. Results of keyframe reconstructions (based purely on matching with 1/30 frames) for some examples in and out of NYU dataset are shown in Figures 4.8, 4.9, 4.10, 4.11, 4.12, and 4.13.

From the results it can be seen that our learned features (particularly the full network version) consistently produce reconstructions close to groundtruth, and are superior to the baselines in terms of both matching accuracy and precision with minimal piecewise constant depth artifacts that occur in some of the baselines when overlapping images have severe appearance distortion. Although in some examples the number of accurate matches is higher in some of the baselines when matching with two frames, the number of accurate matches when matching based on 30 images is consistently higher using our full learned features. This indicates that the matching performance of the baselines is biased towards particular types of appearance changes present in the overlapping images which is an undesirable property.

The only example where a baseline (DenseSIFT) outperforms our learned features with 30 frames is in the TUM 'fr2_xyz' sequence (Example #10 and #12 in Figures 4.11 and 4.13) where no camera rotation is observed. We believe this compromise is a consequence of enforcing our learned features to match across images acquired with a camera undergoing *general* motion during training, where more severe appearance distortion can occur unlike in the stereo or xyz motion case. On the other hand, if we allowed our feature network to learn features specifically for the stereo or xyz motion case, it may have outperformed DenseSIFT in such cases. Nevertheless even in the xyz motion case the

[1]videos showcasing live real-time monocular dense reconstruction are available in this link: https://goo.gl/jUqiBN.

performance of our learned features is not too far behind DenseSIFT and often greatly outperform 7x7 affine patch features and ResNet-50 ImageNet Conv1 features in matching performance.

A Detailed Analysis of Learned Deep Features: Figure 4.14 visualizes the depth map solutions inferred from the cost volume generated using RGB features with that generated using our learned features in some challenging scenarios involving large baseline matching along epipolar lines. Even when only two frames are used for matching, the learned features — consisting of contextual information from a large receptive field — can be successfully matched, while RGB features often struggle to establish a good match as expected. As the number of frames available for matching grows, very realistic and smooth depth maps can be inferred directly from our learned features-based cost volume without the application of any prior. A deeper analysis of the cost volumes generated using the proposed learned features (Figure 4.15) verses those generated using using RGB features highlight three main advantages of using our learned deep features. Particularly, our learned deep features-based cost volume has (i) sharp (non-flat) minima which leads to unambiguous unique matching even in textureless areas without having to rely on priors, (ii) small number of local minima leading to a large basin of convergence and favoring gradient based optimization methods which heavily rely on priors and initialization, and most importantly, (iii) the global minima corresponds to the correct inverse depth solution in majority of the cases, even with only a few number of overlapping frames.

4.4.2 Learned Features as Feature Reconstruction Loss for Unsupervised Single View Depth Estimation

Due to the generalization capabilities of our learned features for matching across different camera models, we now experiment with how learned features for matching trained on NYUv2 dataset can benefit unspervised learning for depth estimation on the KITTI dataset. To do so, we complement the photometric error-based loss function with our learned feature based loss function (Equation 4.8) and learn to regress for the depth map that minimizes the overall loss. An inverse depth smoothness prior similar to Garg et al. [2016] is also employed.

Table 4.4 provides an ablation study of the various experiments carried out. As shown in the *Full-NYUv2* row in the table a combination of using the feature loss with image pairs sampled in both the temporal and spatial (stereo) domain gives us a big improvement over the corresponding baseline methods using only photometric error. In the table we also compare against results of using ResNet-50 ImageNet pre-trained Conv1 features (*ImageNet Feat.*) in the feature reconstruction loss, which produces inferior results. Figure 4.17 which provides a visual comparison of the matching performance of our network against the ResNet-50 Conv1 features may provide an explanation to the depth prediction results obtained by using the respective features in the feature reconstruction loss.

Note that the *Full-NYUv2* experiment relied on our feature prediction network pre-trained on the NYUv2 dataset. However, an interesting question now arises: *Can we let initial depth predictions from our network trained in an unsupervised manner via photometric loss be used as proxy groundtruth to learn features that are good for matching (with randomly initialized weights), and can these learned features for matching in turn improve the original depth predictions when the learned feature matching loss is introduced as before?* Interestingly, this is indeed the case. The results for this experiment is shown the row *KITTI Feat.* in Table 4.4. While the error measures are slightly lower for this

Method	Stereo	Temporal	Feature	Error metric				Accuracy metric		
				Abs Rel	SqRel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Encoder: ResNet50-1by2; Decoder: Bilinear upsampler										
Baseline	✓	✗	✗	0.143	0.859	4.310	0.229	0.802	0.933	0.973
Temporal	✓	✓	✗	0.135	0.905	4.366	0.225	0.818	0.937	0.973
ImageNet Feat.	✓	✗	✓	0.136	0.880	4.390	0.230	0.823	0.935	0.970
KITTI Feat.	✓	✗	✓	0.130	0.860	4.271	0.221	0.831	0.938	0.973
Full KITTI Feat.	✓	✓	✓	0.130	0.904	4.266	0.219	0.831	0.940	0.974
NYUv2 Feat.	✓	✗	✓	0.132	0.906	4.279	0.220	0.831	0.939	0.974
Full-NYUv2	✓	✓	✓	0.128	0.815	4.204	0.216	0.835	0.941	0.975
Encoder: ResNet50-1by2; Decoder: Learnable upsampler										
Baseline2	✓	✗	✗	0.155	1.307	4.560	0.242	0.805	0.928	0.968
Temporal2	✓	✓	✗	0.141	0.998	4.354	0.232	0.814	0.932	0.971
Depth Feat.	✓	✗	✓	0.142	0.956	4.377	0.230	0.817	0.934	0.971
Full-Depth	✓	✓	✓	0.137	0.893	4.348	0.228	0.821	0.935	0.971

Table 4.4: Ablation study on the effect of using different feature types and stereo/monocular sequences during training for single view depth estimation. The result is evaluated in KITTI 2015 using Eigen Split test set, following the evaluation protocol proposed in Godard et al. [2017]. The results are capped at 50m depth. Stereo: stereo pairs are used for training; Temporal: additional temporal pairs are used; Feature: feature reconstruction loss is used. Table reproduced from Zhan et al. [2018].

experiment than *Full-NYUv2*, the accuracy measures are slightly lower than *Full-NYUv2*. The fact that we can keep improving both the features to match and the depth predictions in an alternate fashion, all the while being completely unsupervised, is very promising for future research in this area as it shows that a network can learn to “self-improve”.

Furthermore, instead of using our network architecture for feature predicion, we also use an internal feature of the network, which we call ‘Depth Feat.’ as the features for the feature matching loss. The improvement however is not as significant as using our dedicated network for feature generation. We believe this may be because depth prediction and feature generation are complementary tasks requiring a different set of features, signifying the importance of having a dedicated and specially designed network for feature generation. This however is something that future work should further look into. Figure 4.16 shows the graph of matching error vs inverse depth/disparity for a textureless region in the image (taken from the KITTI dataset) using our learned features (trained on NYUv2) and some other baselines. A clear distinct minimum at the correct inverse depth solution can be observed when using our learned features for stereo matching similar to what what was observed on the NYU-D v2 dataset.

4.5 Future Directions

4.5.1 Learning Feature Matching Confidences

We can associate a confidence c_p^{ϕ} to each pixel in the keyframe image as a measure of how good it is for matching. Note that c_p^{ϕ} is ideally a function of two images, as the confidence is based on factors such as amount of baseline, and amount of occlusion. However, as a compromise we can approximate the matching confidence to be a function of the keyframe image, and this would still give us, for example, the appropriate matching confidence in areas consisting of specular reflections and large textureless regions depending on how well our learned features perform in such regions. Furthermore, depending on the training data, a neural net may learn to predict a low matching confidence for pixels corresponding to objects that are likely to move, based on the keyframe image alone, provided we don't explicitly model dynamic objects.

Traditionally, a common heuristic used to represent matching uncertainty σ_p^{ϕ} is based on the shape of the matching cost function E_{ϕ} at the inverse depth hypothesis ρ_p^{mv} for which the matching cost is minimum (Matthies et al. [1989]; Newcombe [2012]):

$$(\sigma_p^{\phi})^2 \propto \frac{E_{\phi}(\rho_p^{mv})}{\nabla^2 E_{\phi}(\rho_p^{mv})} \tag{4.10}$$

While reasonably accurate, this heuristic is less reliable for measuring uncertainty of specular surfaces, repetitive textures, etc.

In contrast, we aim to use a neural network to learn a mapping from image values to pixelwise matching confidences (inverse of matching uncertainty, σ_p^{ϕ}) that satisfy the following objective:

$$\min_{\sigma_p^{\phi}} \sum_{p\in P} -ln \frac{1}{\sqrt{2\pi}\sigma_p^{\phi}} \exp -\frac{1}{2(\sigma_p^{\phi})^2}(\rho_p - \rho_p^{mv})^2 \tag{4.11}$$

where ρ_p is the groundtruth inverse depth value and $\rho_p^{mv} = \arg\min_{\rho_p} E_{\phi}(\rho_p, \mathbf{f}_{ref}, \mathbf{f}_{live})$ or equivalently $\rho_p^{mv} = \arg\max_{\rho_p^l} P_{\phi}(\rho_p^l, \mathbf{f}_{ref}, \mathbf{f}_{live})$ is the inverse depth value corresponding to the global minimum matching error at pixel location p.

Let us write $c_p^{\phi} = 1/\sigma_p^{\phi}$ to represent the matching confidence at each pixel location p:

$$\min_{c_p^{\phi}} \sum_{p\in P} \frac{(c_p^{\phi})^2}{2}(\rho_p - \rho_p^{mv})^2 - \frac{1}{2}\sum_{p} ln \frac{(c_p^{\phi})^2}{2\pi} \tag{4.12}$$

Simplifying the above expression leads to the following:

$$\min_{c_p^{\phi}} \sum_{p\in P} \frac{(c_p^{\phi})^2}{2}(\rho_p - \rho_p^{mv})^2 - \sum_{i} ln c_p^{\phi} + K \tag{4.13}$$

where $K = -\frac{1}{2}\sum_{p\in P} ln\frac{1}{2\pi}$.

Gradient expressions for Objective (4.13) with respect to c_p^{ϕ} can be easily derived. We can now solve for Objective (4.13) (but summed over all training instances) via Stochastic Gradient Descent (SGD), with lnc_p^{ϕ} regressed at the network output. Note that regressing for lnc_p^{ϕ} instead of c_p^{ϕ} is more desirable as it prevents a negative value being predicted for c_p^{ϕ} and the output as well as the neural network weights will be in a more manageable range.

Normalizing c^{ϕ}: The range of values of the learned confidence map c^{ϕ} may sometimes be very large depending on the nature of the normal inconsistency errors. After the CNN confidence predictor is trained we can simply normalize c^{ϕ} by dividing it with a global constant such that it is in a manageable range. For instance the normalization could be done by: $c_p^{\phi} \leftarrow c_p^{\phi}/\sigma_{c_i^{\phi}}$, where $\sigma_{c_p^{\phi}}$ is the standard deviation of all the predicted c_p^{ϕ}s in the training set. This would make it easier to tune the hyper-parameter λ of our energy formulation for mapping during inference.

Dealing with Occlusion and Dynamic Objects: The presence of occlusion and dynamic objects in the scene (provided they are not explicitly modeled) can lead to errors in mapping. As a heuristic we can set $c_p^{\phi} = 0$ for pixels where the feature error at the global minimum $E_{\phi}(\rho_p^{mv})$ is greater than a certain threshold.

We can now re-write the original matching energy function weighted by the pixelwise learned confidences as follows:

$$E_{\phi}(\boldsymbol{\rho}) = \sum_{p \in P} \sum_{n=1}^{N} \frac{(c_p^{\phi})^2}{N} \left\| \mathbf{f}_{ref}(\mathbf{u}_p) - \mathbf{f}_{live}(\pi(T_{nr}\pi^{-1}(\mathbf{u}_p, \rho_p))) \right\|_1 \tag{4.14}$$

This confidence weighting will increase the robustness of the inverse depth solution to potentially incorrect matches.

Note that here we have made the crucial assumption that errors in inverse depths are directly proportional to (a linear function of) re-projection errors in pixel space as well as learned feature error space, which is approximately true in the case of low-baseline keyframe reconstruction/camera tracking, and given that we have explicitly trained our learned feature matching error distribution along epipolar lines to follow a Gaussian distribution. Hence the reason (a linearly scaled version of) our learned c_p^{ϕ} can be directly used as a weighting for the observation model computing re-projection error in pixel/feature space. While this weighting is more valid for low-baseline mapping/camera tracking, it may also be used in the observation model for bundle adjustment.

Our proposed method can be used to learn the matching confidence as a function of the keyframe image for any given feature extractor by simply replacing $\mathbf{f}_{ref}$ and $\mathbf{f}_{live}$ with the desired features in Objective 4.11.

4.5.2 A Convex Approximation to the Multi-View Observation Model for Keyframe Reconstruction

As the shape of the matching cost function $E_{\phi}(\rho_p, \mathbf{f}_{ref}, \mathbf{f}_{live})$ for a keyframe pixel is typically well modeled with a Gaussian function at the global minimum ρ_p^{mv} (due to the unique matching enabled by using our learned features), we can approximate $E_{\phi}(\rho_p, \mathbf{f}_{ref}, \mathbf{f}_{live})$ with a Gaussian function with learned c_p^{ϕ} (following up from the previous section), resulting in a convex approximation to the matching error function:

$$E_{\phi}(\boldsymbol{\rho}) \approx \sum_{p \in P} (c_p^{\phi})^2 (\rho_p - \rho_p^{mv})^2 \tag{4.15}$$

This convex approximation will likely lead to much faster convergence at a slight sacrifice in reconstruction accuracy and allow for convergence guarantees to be made. Note that in order to obtain a more refined ρ_p^{mv} based on a *discrete* cost volume, we can perform a single Newton step update on the discrete inverse depth values at the minima locations, i.e. $\rho_p^{mv} = \rho_p^{mv} - \frac{\nabla E_\phi(\rho_p^{mv})}{\nabla^2 E_\phi(\rho_p^{mv})}$.

4.5.3 Introducing Additional Features for Matching

Our experiments were restricted to having a single data term for enforcing feature consistency, as our network was implicitly allowed to learn a combination of low and high level features that are suitable for matching. Nevertheless, it may be beneficial to introduce additional data terms for improved matching reliability. For this purpose we can re-write Equation 3.3 in a more generic form as follows:

$$E(\boldsymbol{\rho}) = \sum_{p \in P} \sum_{m} \frac{1}{\lambda^m} E_{data}^m(\rho_p) + E_{prior}(\rho_p). \tag{4.16}$$

In this instance, E_{data}^1 could be based on our learned features from this chapter, and the features for each additional E_{data}^m could be, for example, low-level RGB values (computing photometric error), learned surface normal/depth-based warped patches (i.e. more accurately warped patches based on relative camera pose and learned surface normals and depths), and on the other extreme, features trained for high-level tasks such as detection, classification, and semantic segmentation. Introducing high level features would be useful for instance if the epipolar line passes through two classes (e.g. table and chair), in which case we could match based on knowledge of class even though the pixels themselves look visually similar.

The downside in using the highest level features however is that errors in class label prediction will propagate into the matching process. Furthermore, if we were to match based on highest-level features alone, we might not be able to match with sub-pixel precision as, intuitively, these features are not likely to contain detailed low-level image features which are needed for matching within a region belonging to a certain class.

The experiments that led to the design choice of our proposed network architecture suggest that (robust) low-level feature matching is always useful to complement large receptive field (potentially high-level) feature matching. Having an additional data term based purely on pixel color (ideally photometrically calibrated) will reduce potential loss in matching precision due to severe perspective distortion (although the learned features by our method are trained to be invariant to perspective distortion). Having multiple additional data terms will complicate the parameter search for the best λ_ms and therefore an end-to-end learning scheme that builds on our network architecture would be helpful.

4.5.4 Learned Features for Dense Alignment-Based Camera Tracking

Our learned features are likely to benefit direct alignment based camera tracking as well. Dense tracking using ImageNet features have already shown to enable robust tracking in the presence of severe appearance changes between images in Czarnowski et al. [2017]. In Schmidt et al. [2017], learned features from their FCN based network enabled robust model to frame alignment in the presence of severe appearance changes. In Wang et al. [2017] learned features are used for aligning patches in two images based on the Inverse Compositional Lucas & Kanade (IC-LK) dense image (feature) alignment method (similar to what is proposed below, but with no knowledge of the map), for the purpose of object tracking.

For completeness, we here detail the efficient inverse compositional dense image (feature) alignment method (Baker and Matthews [2004]) for camera pose estimation, leaving experimentation for future work. The goal is to minimize the following expression, at each iteration:

$$\sum_{p\in P}(c_p^{\phi})^2[\mathbf{f}_{ref}(W(p,\Delta\xi)) - \mathbf{f}_{live}(W(p,\xi))]^2 \tag{4.17}$$

with respect to $\Delta\xi$, the change in pose, that will incrementally align the keyframe and live images. The optimization process can stop when $\Delta\xi < \epsilon$. $\xi \in \mathfrak{se}(3)$ is the corresponding Lie algebra of the Lie group pose $T \in \mathbf{SE}(3)$.

It is important to note that in Objective 4.17, c_p^{ϕ} is the *learned* pixel-wise matching confidence using our method proposed in Section 4.5.1. $W(p,\xi)$ is a pixel warp operation such that $W(p,\xi) = \pi(T\pi^{-1}(\mathbf{u}_p,\rho_p))$ and it is updated after each iteration as follows:

$$W(p,\xi) \leftarrow W(p,\xi) \circ W(p,\Delta\xi)^{-1} \tag{4.18}$$

Linearizing the reference feature map $\mathbf{f}_{ref}$ around identity pose ($\Delta\xi = 0$), using the first-order Taylor expansion, we obtain:

$$\sum_{p\in P}(c_p^{\phi})^2[\mathbf{f}_{ref}(W(p,0)) + \nabla\mathbf{f}_{ref}\frac{\partial W}{\partial\xi}\Delta\xi - \mathbf{f}_{live}(W(p,\xi))]^2 \tag{4.19}$$

where $W(p,0) = p$ is the identity warp. The Jacobian $\nabla\mathbf{f}_{ref}\frac{\partial W}{\partial\xi}$ is evaluated $\forall p$.

The solution for $\Delta\xi$ that minimizes the Expression 4.19 is:

$$\Delta\xi = H^{-1}\sum_{p\in P}[\nabla\mathbf{f}_{ref}\frac{\partial W}{\partial\xi}]^T(c_p^{\phi})^2[\mathbf{f}_{live}(W(p;\xi)) - \mathbf{f}_{ref}(p)] \tag{4.20}$$

where H is the n x n (Gauss-Newton approximation to the) Hessian matrix (with $n = 6$ being the number of parameters to estimate in $\Delta\xi$) and is given by:

$$H = \sum_{p\in P}[\nabla\mathbf{f}_{ref}\frac{\partial W}{\partial\xi}]^T(c_p^{\phi})^2[\nabla\mathbf{f}_{ref}\frac{\partial W}{\partial\xi}] \tag{4.21}$$

The Jacobian $\nabla\mathbf{f}_{ref}\frac{\partial W}{\partial\xi}$ turns out to be a 32 x 6 dimensional matrix (as 32 is the dimensionality of our feature vector for each pixel), and is given by:

$$\nabla\mathbf{f}_{ref}\frac{\partial W}{\partial\xi} = \begin{pmatrix}\nabla\mathbf{f}_{ref,x} & \nabla\mathbf{f}_{ref,y}\end{pmatrix}\begin{pmatrix} f_x\rho_p & 0 & -f_x x_p\rho_p^2 & -f_x x_p y_p\rho_p^2 & f_x(1+x_p^2\rho_p^2) & -f_x y_p\rho_p \\ 0 & f_y\rho_p & -f_y y_p\rho_p^2 & -f_y(1+y_p^2\rho_p^2) & f_y x_p y_p\rho_p^2 & f_y x_p\rho_p \end{pmatrix} \tag{4.22}$$

where $(x_p, y_p, 1/\rho_p)$ is the corresponding 3D point of pixel $\mathbf{u}_p$ in the keyframe image, and f_x and f_y are the focal lengths.

Since the Hessian matrix H is independent of the warp parameters ξ, it is constant across iterations, and therefore needs to be computed only once. This is in contrast to the Lucas Kanade alignment method (Lucas et al. [1981]) that requires computation of the Hessian after every iteration. Based on the taxonomy in Baker and Matthews [2004], the Lucas Kanade alignment method falls into the less efficient forward additive iterative alignment type, and a detailed comparison can be found in their paper.

While our learned c_p^{ϕ} can attenuate most of the outlier matches, other robust functions can be used in place of the squared L2 loss in Expression 4.17. When a robust norm like L1 or Huber is used, we can for instance switch to the re-weighted iterative least squares-based optimization method. In order to make the tracking robust to occlusions and dynamic objects, similar to DTAM (Newcombe et al. [2011b]) we can remove the associated pixels from the tracking process by checking if the feature error is greater than a certain threshold. This threshold will need be dynamically adjusted at each iteration step.

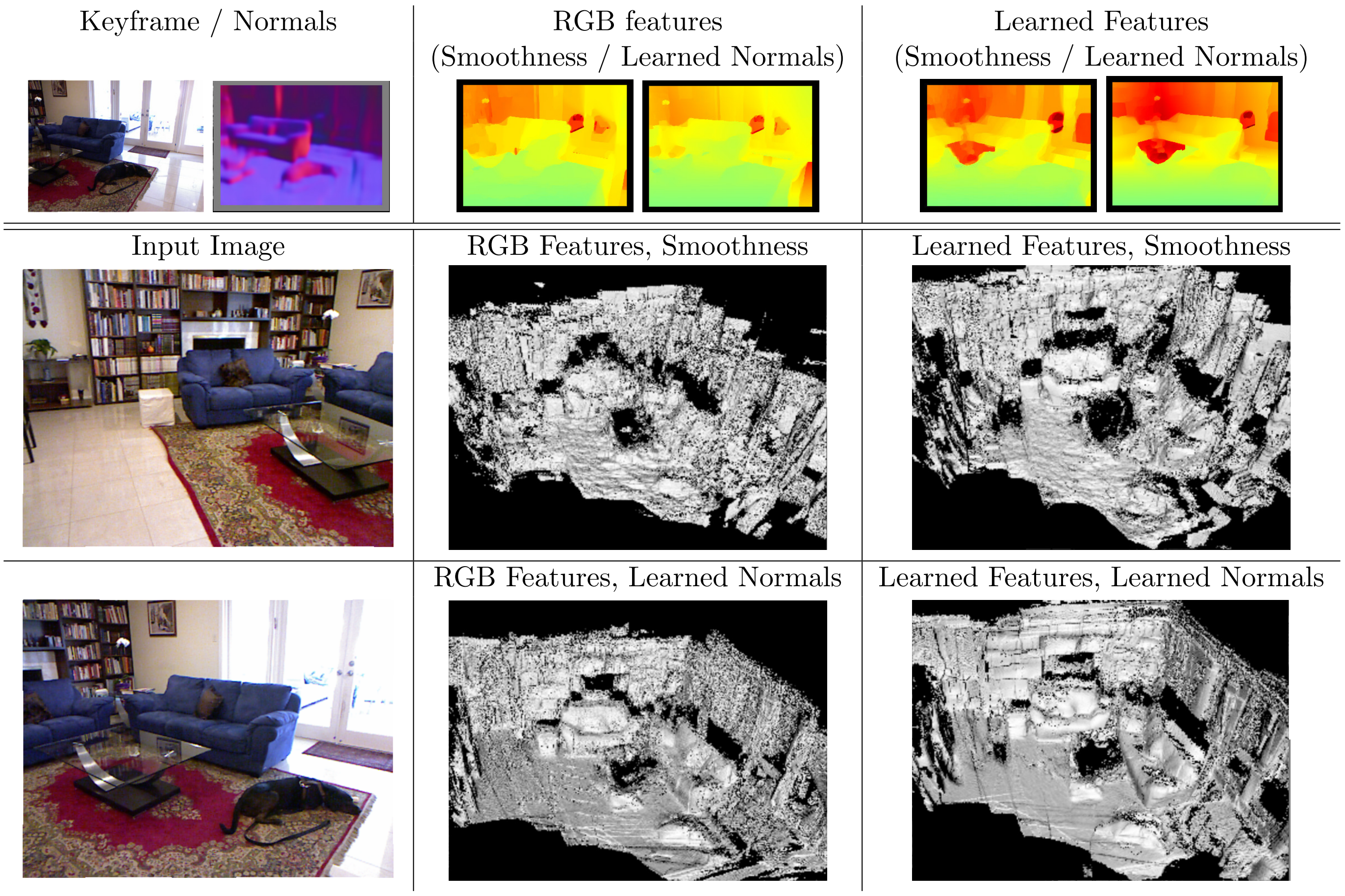
Note that in the dense tracking framework of DTAM (Newcombe et al. [2011b]) a coarse to fine pose estimation and initialization strategy is employed, where relative pose is estimated starting from the coarsest scale, with the estimated pose coming from the coarser scale used as initialization. We could also employ such a strategy in our case as our network explicitly produces feature maps that are good for matching at 4 different scales.

4.5.5 Learned Features for Other Computer Vision Tasks

Just as ImageNet pre-trained features have been successfully used as model initialization for various other computer vision tasks like depth and surface normal prediction (Eigen and Fergus [2015]; Kendall et al. [2015]; Laina et al. [2016]), it is interesting to explore the use our features that are good for matching for other vision tasks such as semantic segmentation, image retrieval, object detection and tracking, and even depth and surface normal prediction. Transfer learning is mainly useful when the task at hand does not consist of a lot of manually annotated labels and in order to prevent over-fitting. In Figures 4.18 and 4.19 we provide a visualization of the 32-channel output of the fast and full versions of our network, in order to visualize what sort of feature maps our network has implicitly learned to produce.

4.6 Conclusion

In this chapter we presented a novel Convolutional Neural Network architecture and a method for learning context-aware deep features which are "good features to match", in the context of dense mapping. We presented an extensive visual analysis which highlight some of the desirable properties of these deep features (invariance to illumination and viewpoint changes and ability to uniquely match thanks to the large receptive field). With the help of extensive quantitative analysis on three different datasets it was shown that our learned features generalize well across data captured with different cameras to give state-of-the-art reconstructions of indoor scenes in real-time. Initial experiments show promising stereo matching results even in substantially different outdoor scenes in the KITTI dataset where not only the camera but also the scene brightness and contents differ substantially. This generalizability enabled our learned features to be used in a feature reconstruction loss for unsupervised depth estimation in the KITTI dataset, improving upon depth estimation results over a pure photometric error-based loss. We discussed some interesting avenues for future work which include learning matching confidences and looking at whether our learned features are useful for tasks other than matching (semantic segmentation and object detection for instance). Beyond dense mapping, our learned features are likely to enable more robust camera tracking when simply used in place of image color in a direct image registration framework.



Keyframe / Normals
RGB features (Smoothness / Learned Normals)
Learned Features (Smoothness / Learned Normals)
Input Image
RGB Features, Smoothness
Learned Features, Smoothness
RGB Features, Learned Normals
Learned Features, Learned Normals

Figure 4.5: A snapshot of our results on the 'living_room_0075' sequence from the NYU-D V2 raw dataset. The **Top part** of the figure shows a sample frame from the sequence selected as a keyframe in our SLAM framework and the depth maps estimated for this keyframe with different loss combinations. From left to right: (1) Keyframe, (2) the estimated normal map for this keyframe (by Eigen and Fergus [2015]), and from then onwards we show the depth maps generated using: (3) RGB features and inverse depth smoothness prior (result of Newcombe et al. [2011b]), (4) RGB features and learned normals-based smoothness prior (result of Chapter 3), (5) learned features and inverse smoothness prior, and (6) learned features and learned normals-based smoothness prior. Note how using learned features drastically improves the depth map estimate for the keyframe. In particular notice the back of fully visible sofa, the wall and glass behind the sofa and the floor underneath. The reconstructions corresponding to these regions get visibly better as the wrongly estimated surfaces and depth discontinuities are fixed when our learned features replace RGB features in cost volume estimation. In the **Bottom part** of the figure, the first column shows a few sample images of the room. The next two columns show the results obtained by fusing the depth maps for the keyframes using RGB and learned features respectively. As expected, learned features help produce consistent and accurate depth maps for each keyframe allowing easy fusion to get high quality dense volumetric maps. The learned normals-based smoothness prior additionally removes the noise in the reconstructions.

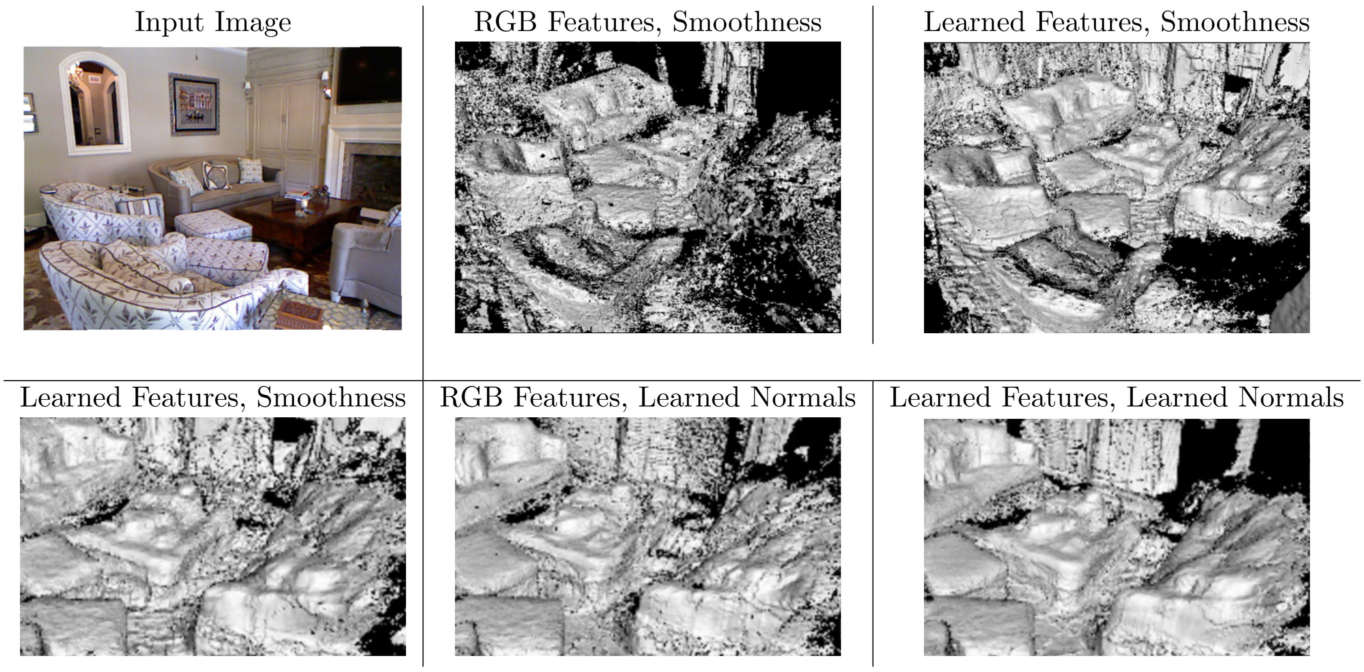

Input Image
RGB Features, Smoothness
Learned Features, Smoothness
Learned Features, Smoothness
RGB Features, Learned Normals
Learned Features, Learned Normals

Figure 4.6: Qualitative comparison of reconstructions obtained by our mono SLAM framework for the sequence ‘living_room_0080’ against those of baselines. While a standard SLAM framework like Newcombe et al. [2011b] gives inconsistent depth maps across keyframes that cannot be fused and thus leave holes in the reconstruction (top-middle), learned features are more accurately matched. The bottom part of the figure shows how using both our learned features and the learned normals-based smoothing of Chapter 3 improves the quality of fused maps.

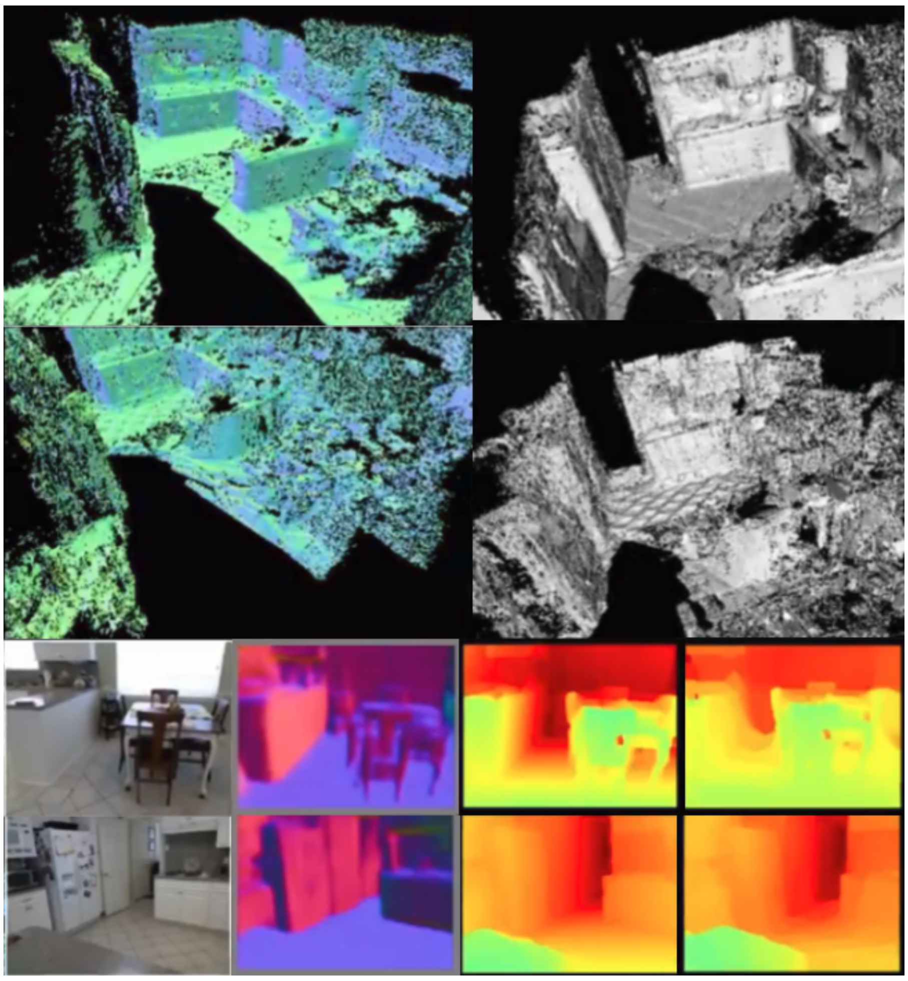


Figure 4.7: Qualitative comparison of reconstructions obtained by our mono SLAM framework for the sequence 'kitchen_0046' against those of DTAM. The top row shows our fused reconstruction using both our learned features for matching and learned normals-based smoothness prior. The next row shows the reconstruction obtained using RGB features for matching and the inverse depth smoothness prior (result of DTAM). Note how the inconsistencies in the depth maps cause the latter reconstruction to be noisy. The bottom part of the figure shows two of the keyframes in the sequence, along with their corresponding estimated normals (Eigen and Fergus [2015]), and estimated depth maps using our learned features and learned normals-based smoothness prior (left) and RGB features and inverse depth smoothness prior (right).

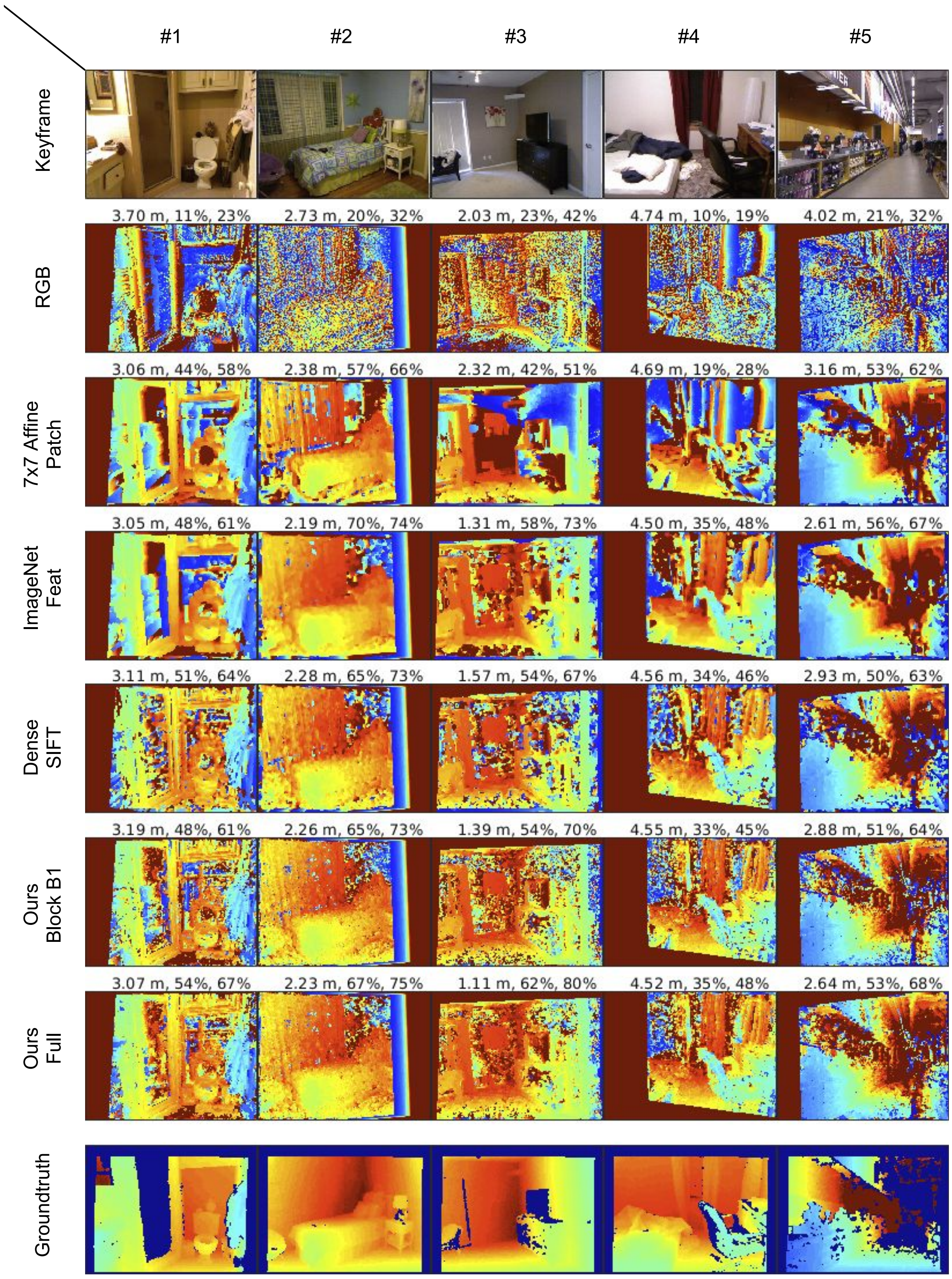


Figure 4.8: Comparison of keyframe reconstructions (based purely on matching with 1 frame) using our learned features against those of baselines (example set 1). The three numbers above each result denote RMSE (m), ACCURACY (with threshold set to 1.1 — measuring highest precision matches), and ACCURACY (with threshold set to 1.25).

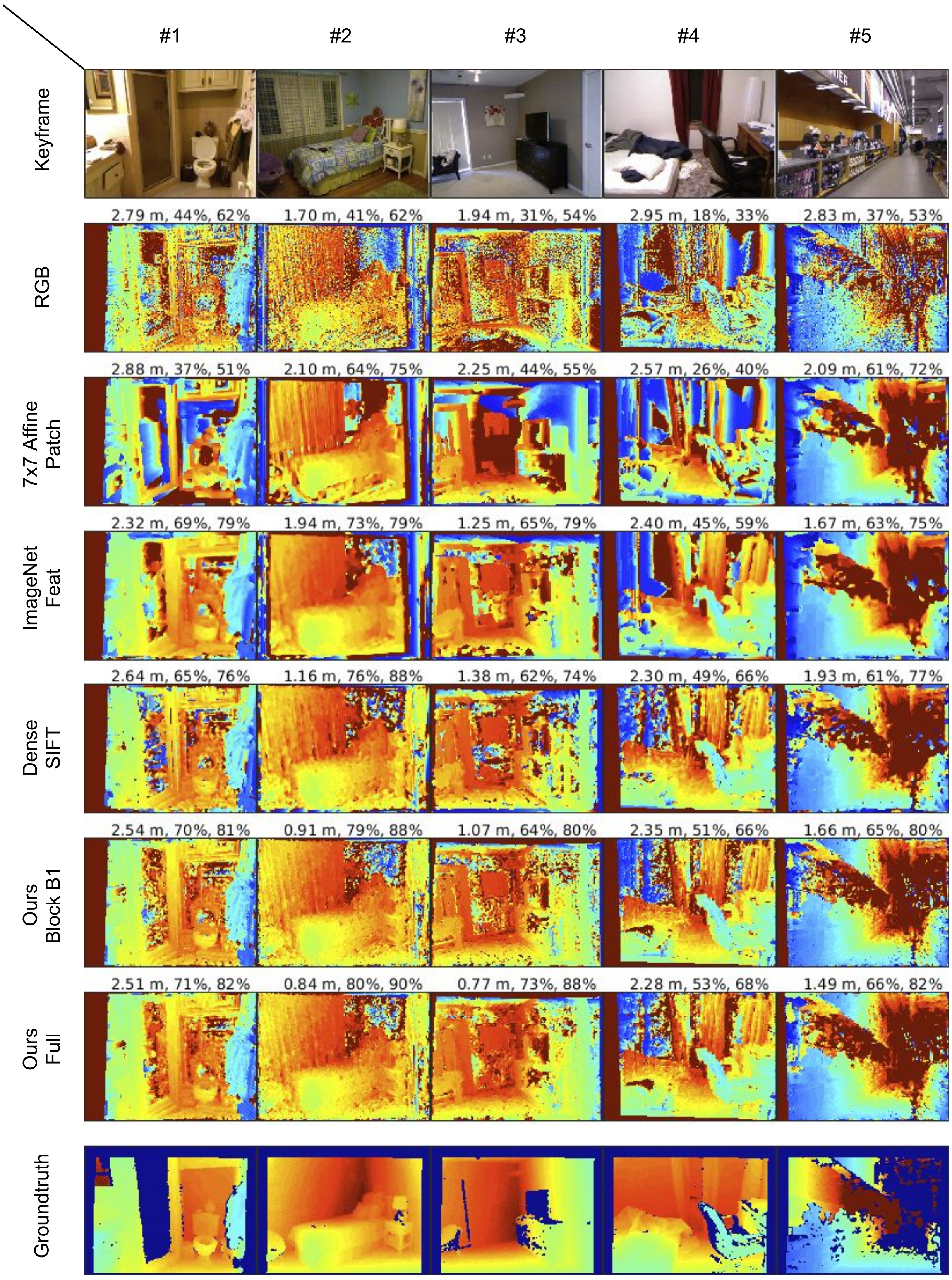


Figure 4.9: Comparison of keyframe reconstructions (based purely on matching with 30 frames) using our learned features against those of baselines (example set 1). The three numbers above each result denote RMSE (m), ACCURACY (with threshold set to 1.1 — measuring highest precision matches), and ACCURACY (with threshold set to 1.25).

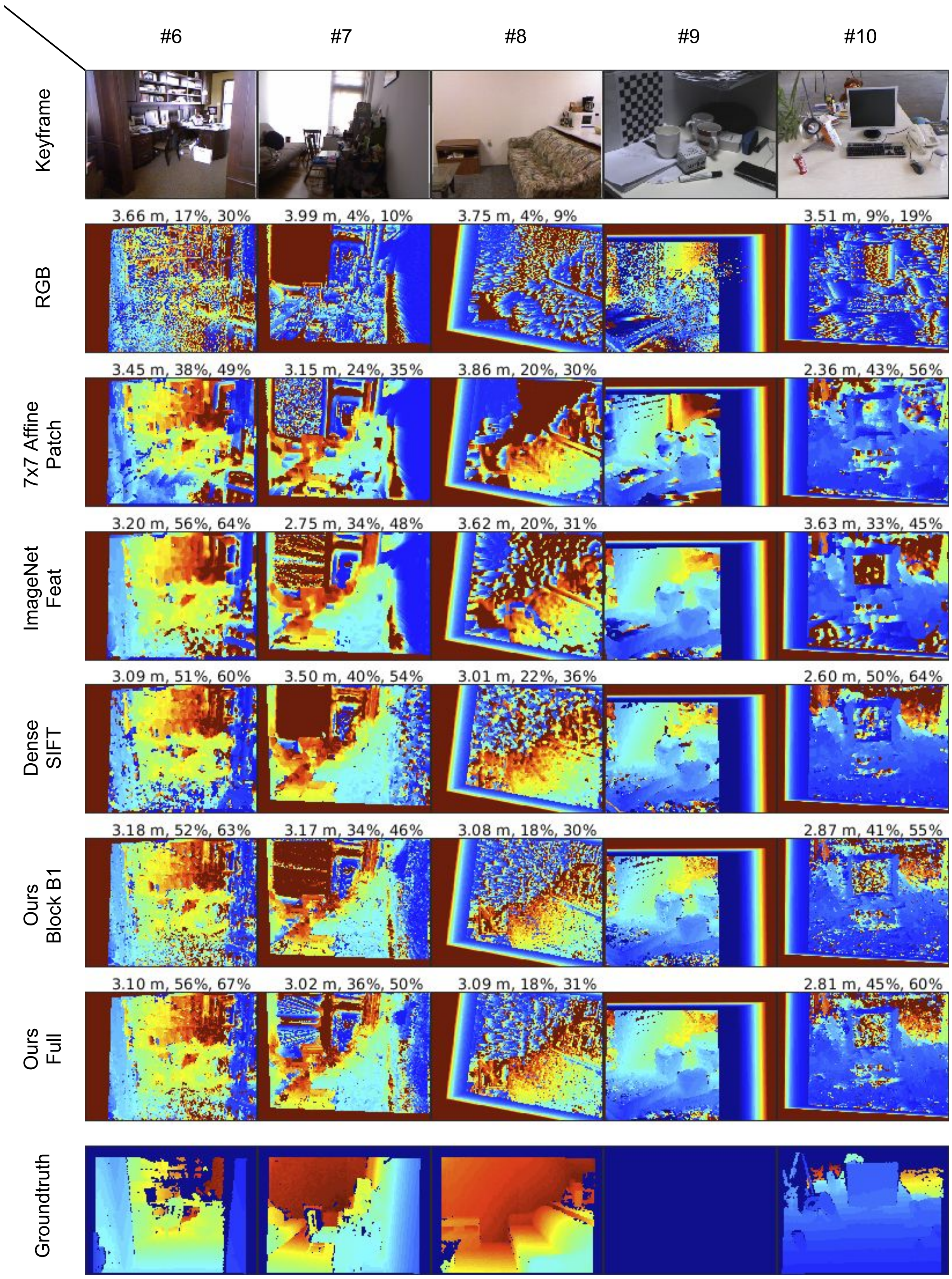


Figure 4.10: Comparison of keyframe reconstructions (based purely on matching with 1 frame) using our learned features against those of baselines (example set 2). The three numbers above each result denote RMSE (m), ACCURACY (with threshold set to 1.1 — measuring highest precision matches), and ACCURACY (with threshold set to 1.25).

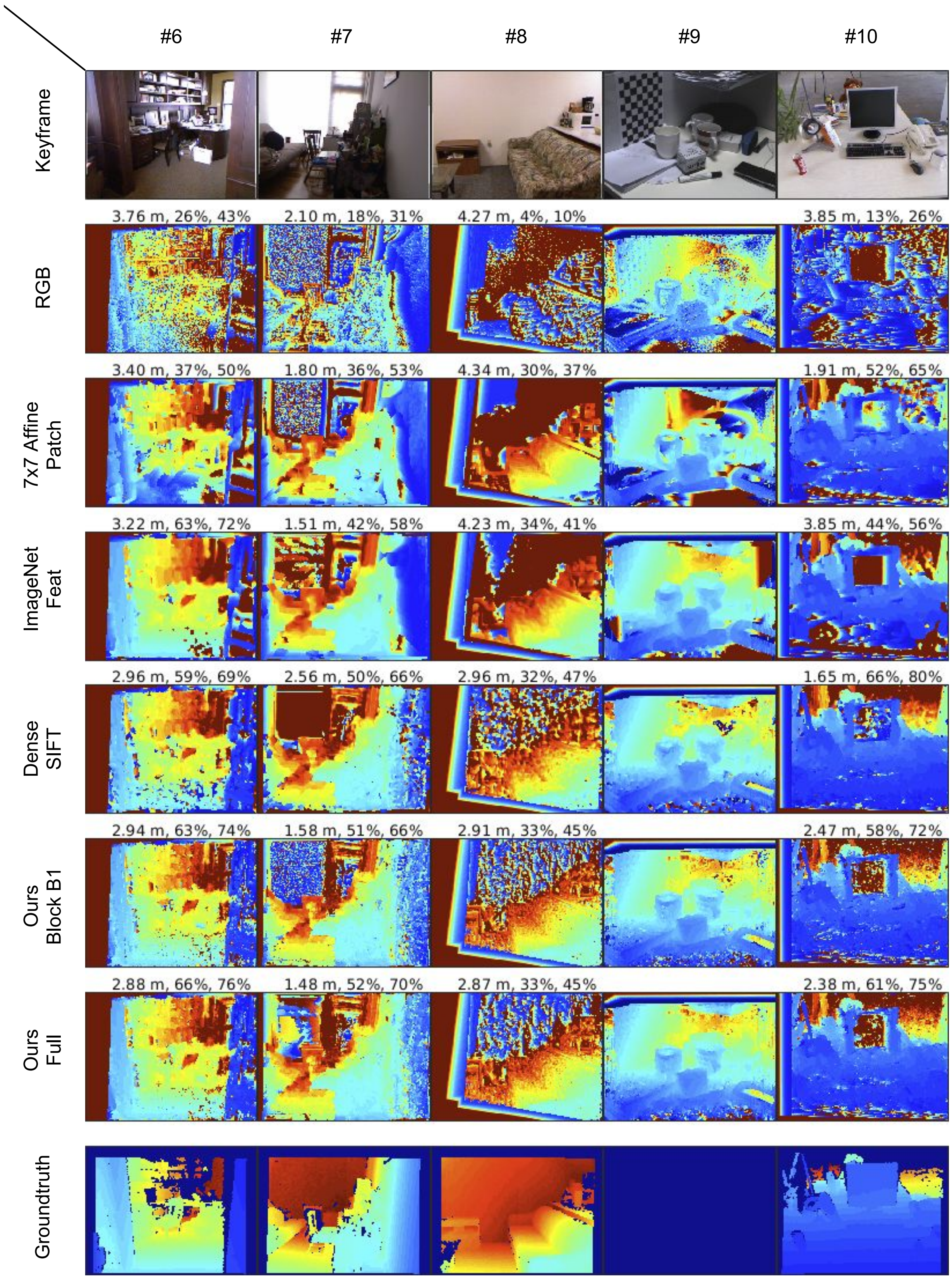


Figure 4.11: Comparison of keyframe reconstructions (based purely on matching with 30 frames) using our learned features against those of baselines (example set 2). The three numbers above each result denote RMSE (m), ACCURACY (with threshold set to 1.1 — measuring highest precision matches), and ACCURACY (with threshold set to 1.25).

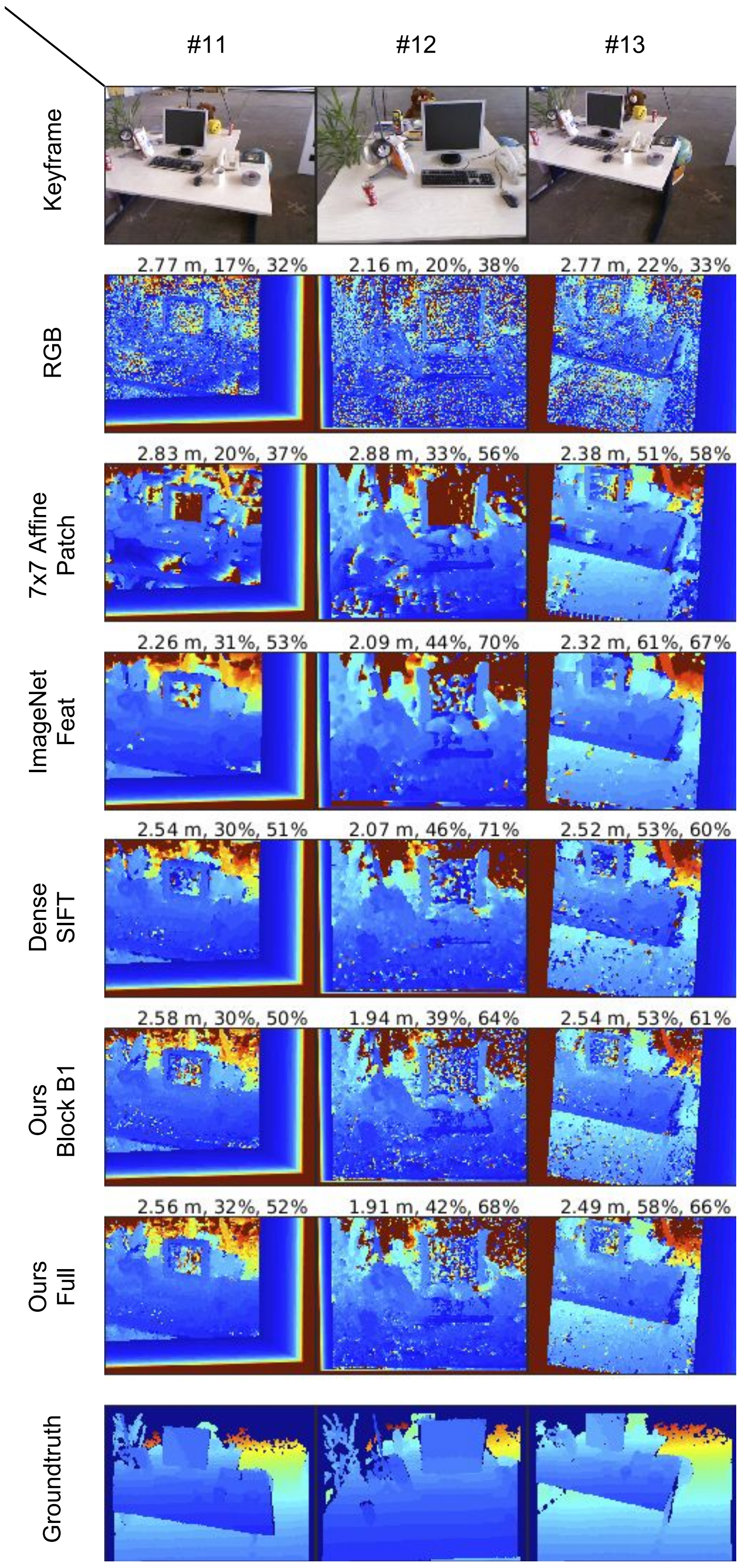


Figure 4.12: Comparison of keyframe reconstructions (based purely on matching with 1 frame) using our learned features against those of baselines (example set 3). The three numbers above each result denote RMSE (m), ACCURACY (with threshold set to 1.1 — measuring highest precision matches), and ACCURACY (with threshold set to 1.25).

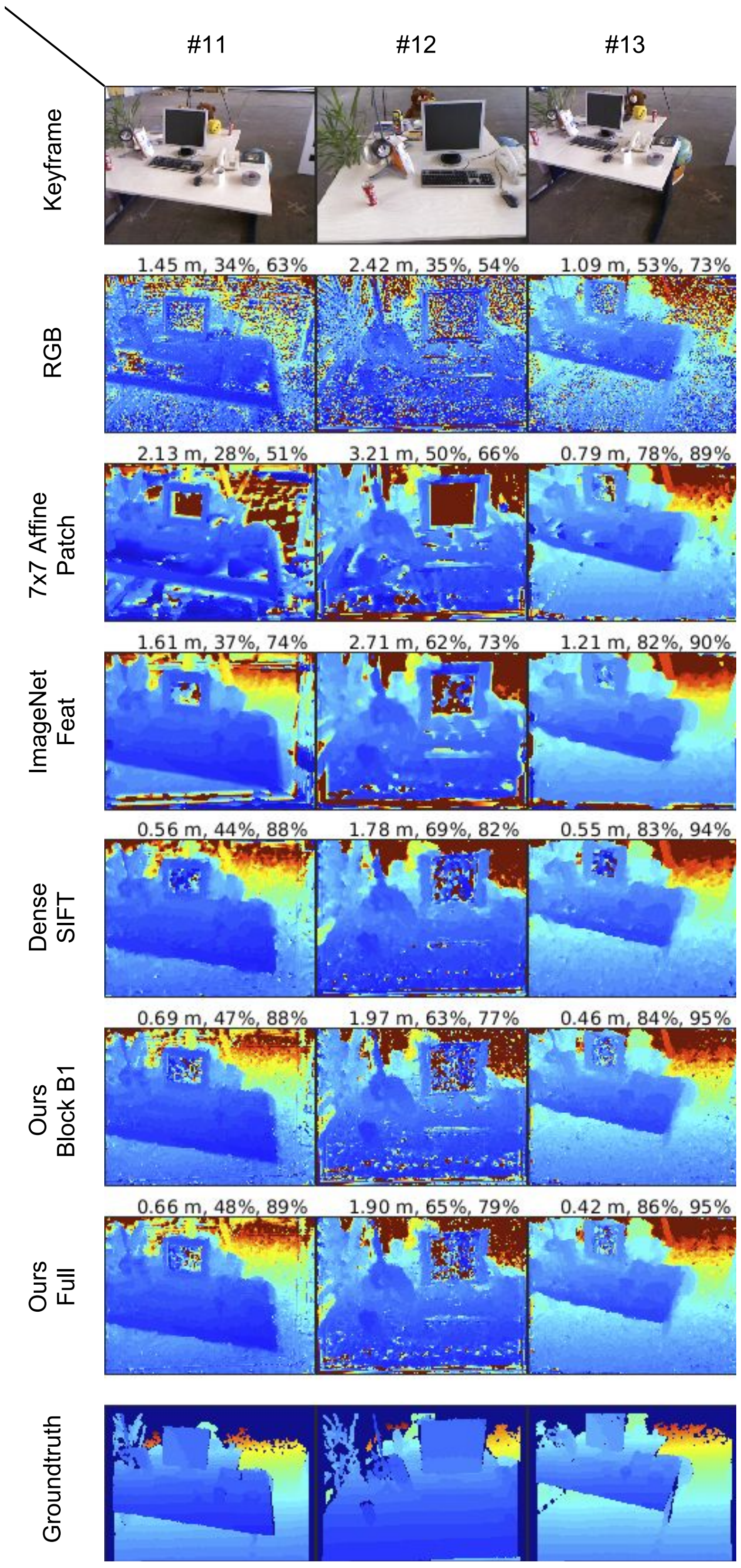


Figure 4.13: Comparison of keyframe reconstructions (based purely on matching with 30 frames) using our learned features against those of baselines (example set 3). The three numbers above each result denote RMSE (m), ACCURACY (with threshold set to 1.1 — measuring highest precision matches), and ACCURACY (with threshold set to 1.25).

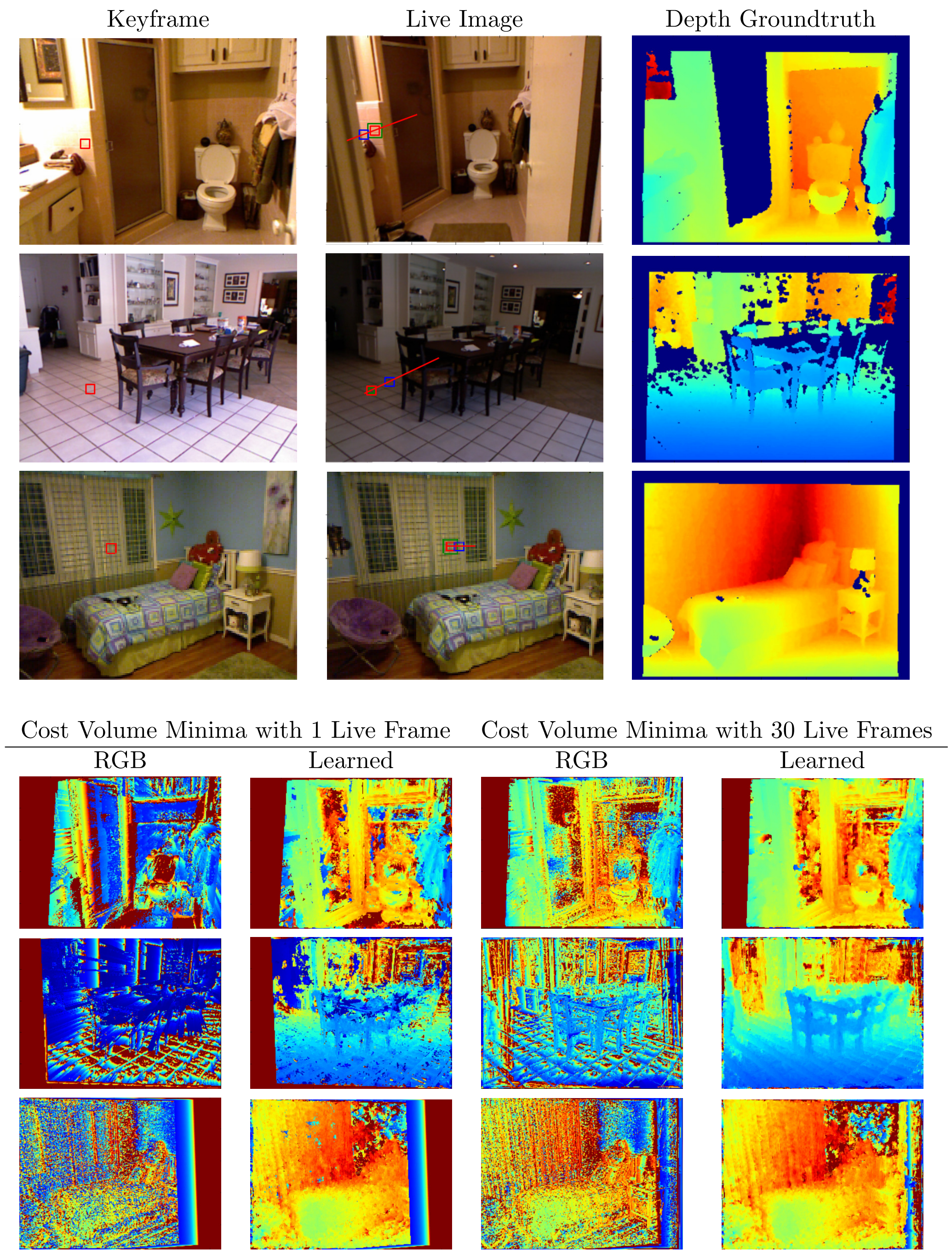

Keyframe
Live Image
Depth Groundtruth
Cost Volume Minima with 1 Live Frame
Cost Volume Minima with 30 Live Frames
RGB
Learned
RGB
Learned

Figure 4.14: Figure compares RGB and learned features for keyframe reconstruction purely based on feature matching for: a bathroom, a dining room, and a bedroom sequence from the NYU-D v2 dataset. **Top:** The red square in the first image in each row is a point of interest, which has to be mapped to the corresponding image in the second column on the epipolar line (denoted in red). The red square in the second image correspond to the predicted match using deep features which aligns correctly with the true match (green square) while the blue square represents the best RGB match. The last column is the ground truth depth map for each case. **Bottom:** The left column shows the depth maps obtained by matching the two frames shown in the top using RGB and learned features respectively. It is evident that the learned features are more reliable for matching two frames and successfully match far more points correctly despite absence of local texture. The right column shows the depth maps obtained by minimizing RGB-based and learned features-based cost volumes respectively given 30 live frames. Matching evidence gets accumulated over frames for both cases but the cost volume constructed using the learned features gives considerably better results.

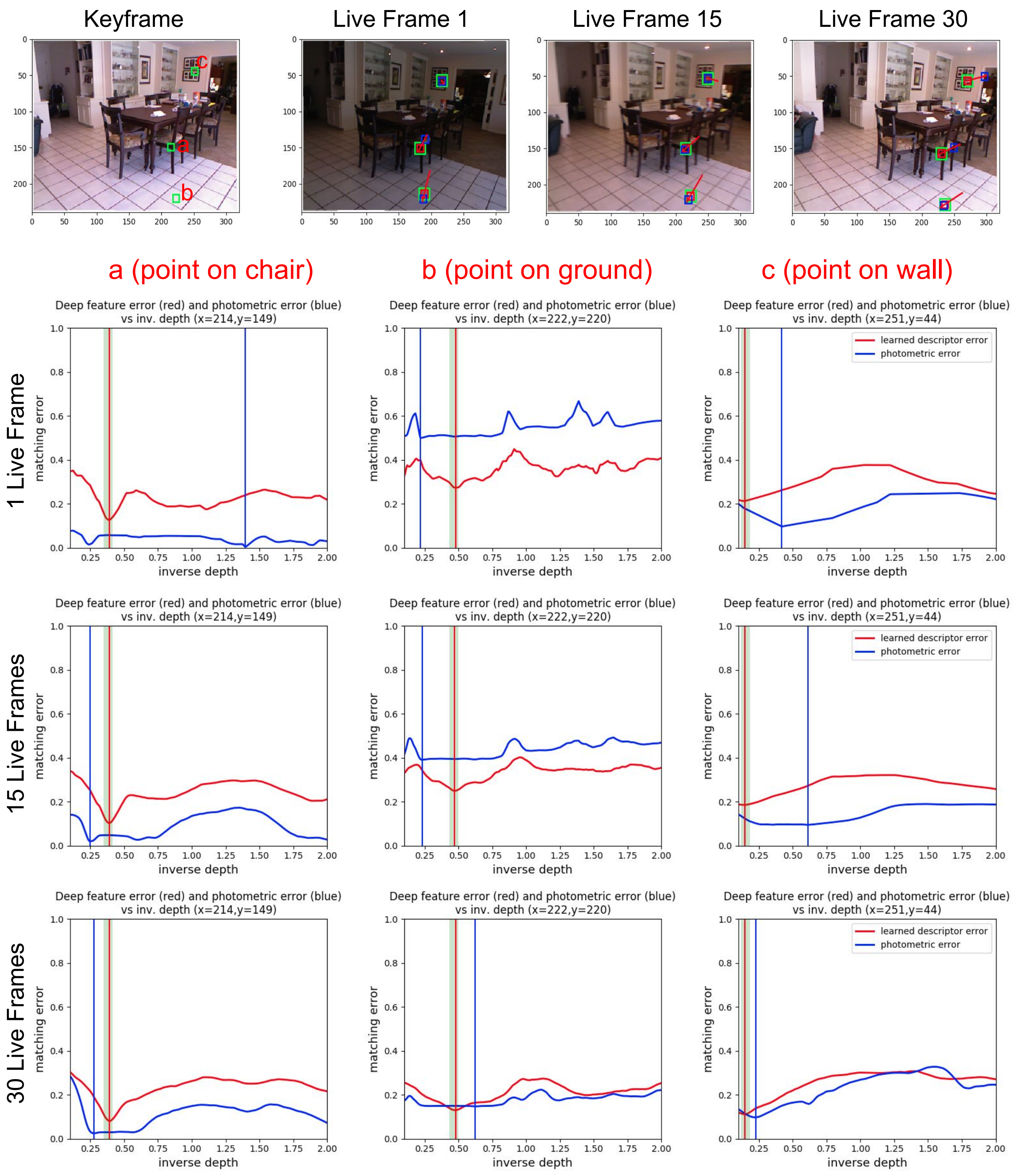


Figure 4.15: Figure shows the cost-volume as it evolves after accumulating evidences from multiple frames for 3 different points, using learned features (red) and rgb features (blue). The red and blue vertical lines in the plots indicate the location of the minimum using learned features and rgb features respectively, and the green vertical line indicates the location of the groundtruth minimum. From top to bottom, the plots correspond to cost volumes accumulated over 1, 15 and 30 live frames respectively. In the top images, the green squares show the location of the 3 points of interest in the reference image (left), and the corresponding groundtruth locations in 3 of the live images that were used to generate the cost volume. The red and blue squares show the matched locations of the same 3 points after searching along the corresponding epipolar lines in the live frames using learned features and rgb features respectively.

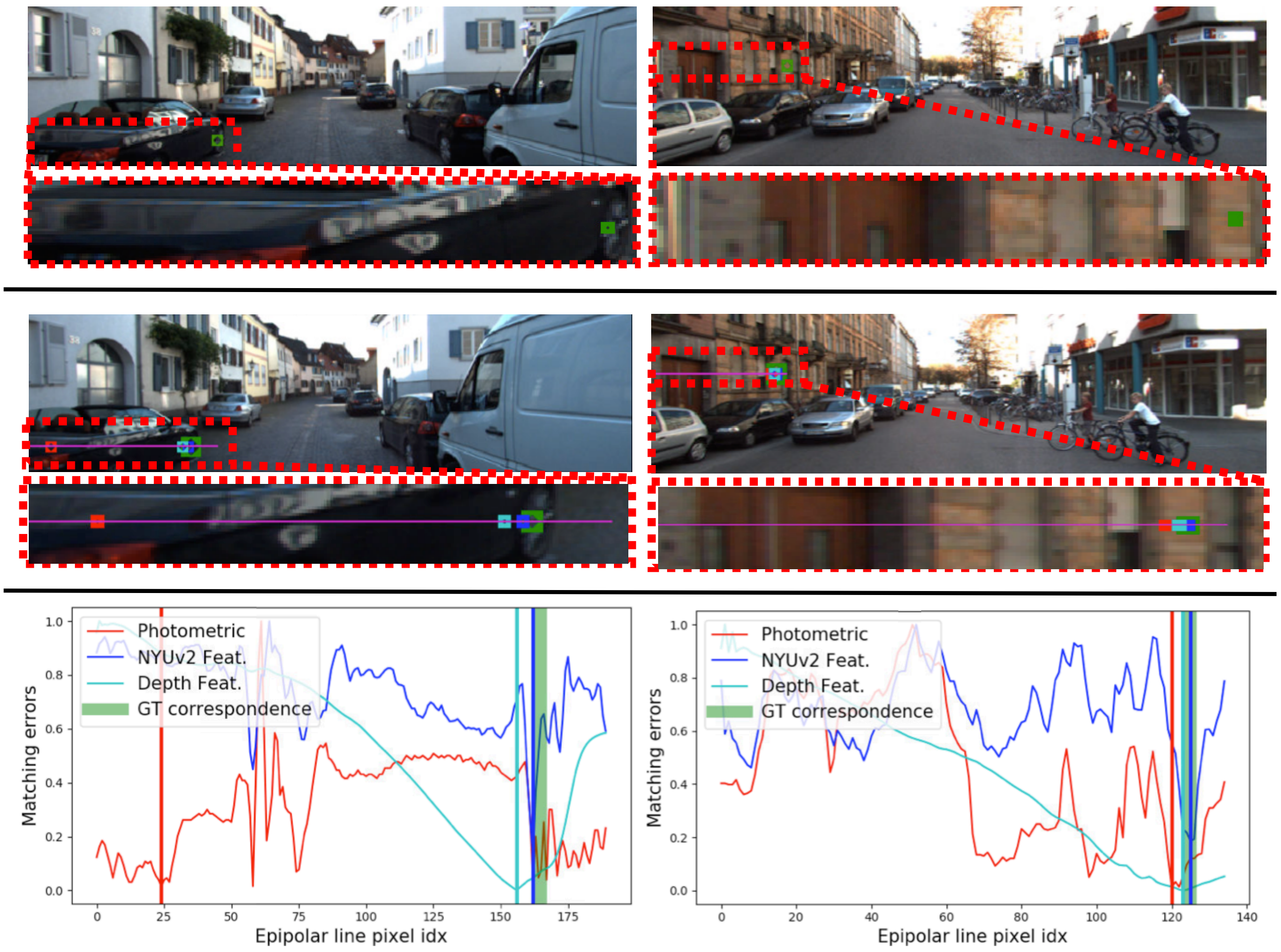


Figure 4.16: Visual ablation study on the effect of using different feature types for stereo matching in the KITTI dataset. Rows: (1) Left image; (2) Right image; (3) Matching error using color intensity and deep features. Photometric loss is not robust when compared with feature loss, especially in ambiguous regions. Figure reproduced from Zhan et al. [2018].

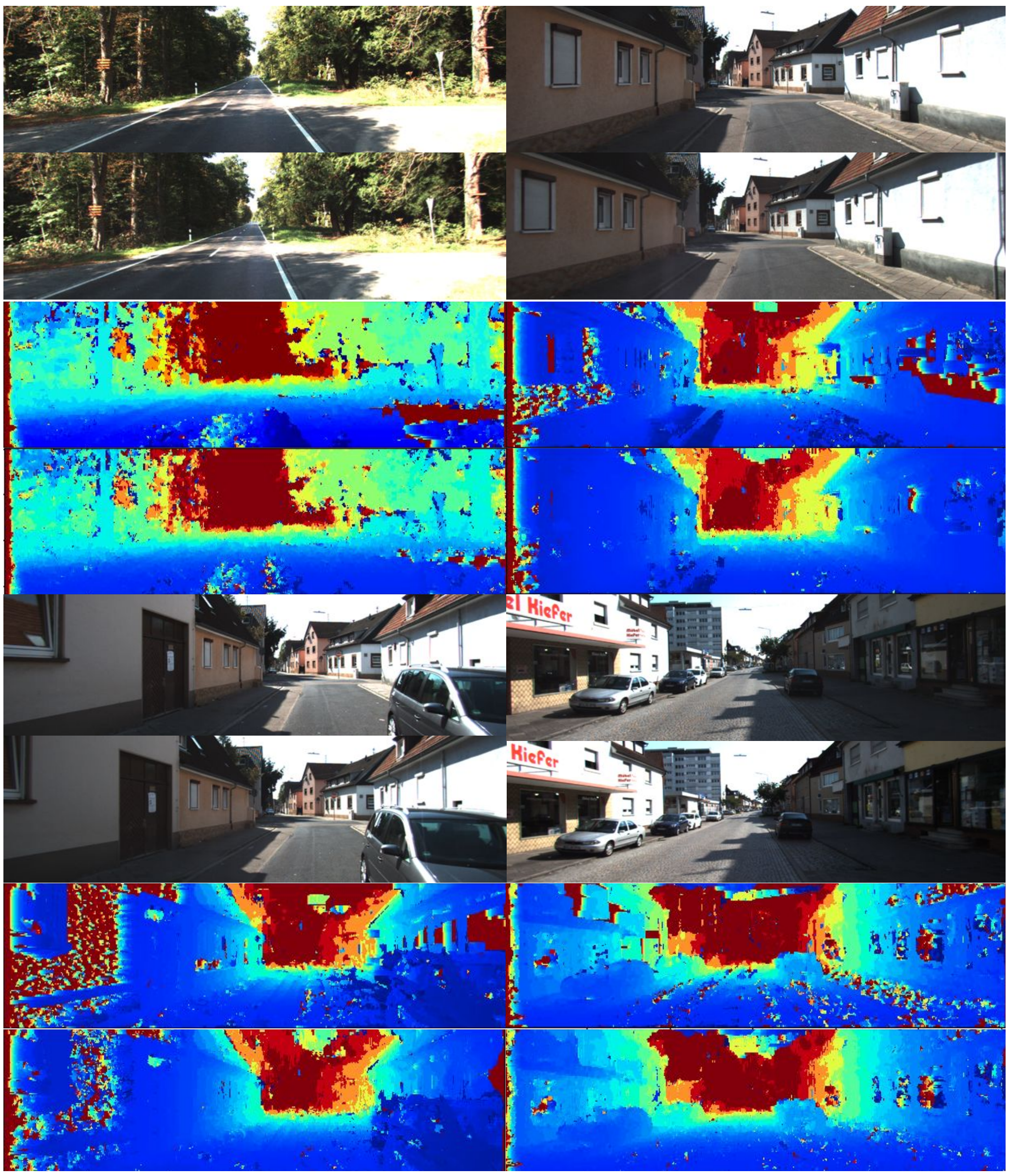


Figure 4.17: Comparison of the performance of low-level features trained for image classification, for pixel matching, against that of our learned features. From top to bottom in each example are left and right images in the stereo pairs obtained from the KITTI dataset, followed by the depth maps obtained via matching with *conv*1 features from $ResNet50\_1by2$ (ImageNet Feat.), and matching with our learned features respectively. Notice again how the larger receptive field enables matching of large textureless regions. Nevertheless, this goes to show that pre-trained features for other vision tasks can form a decent alternative for the purpose of dense matching (at least in the stereo case). It is important to note however that ImageNet is a *manually* annotated dataset, while our features are self-supervised from RGB-D data, and can even be learned unsupervised, given depth from unsupervised depth estimation as proxy groundtruth.

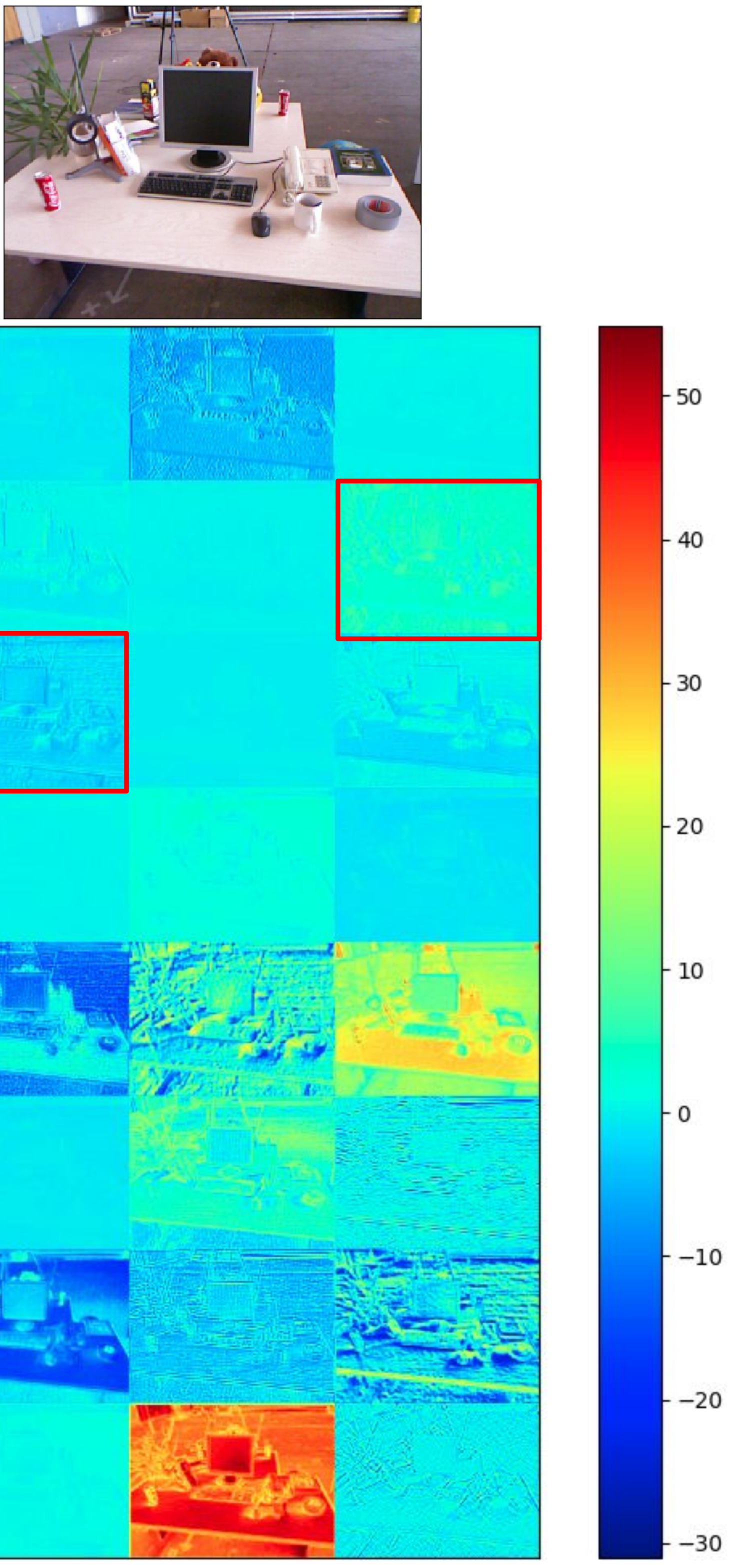
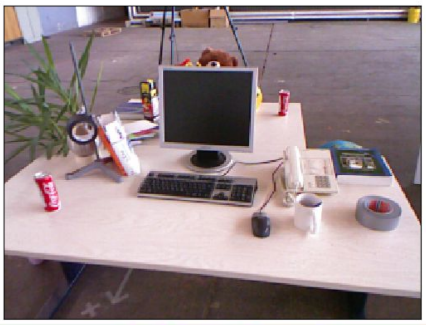


Figure 4.18: Output of the fast version of our learned matching feature network (only Block B1). Interestingly most of the difference in the output of the fast and full versions of our network are concentrated in the feature maps highlighted by the red boxes.



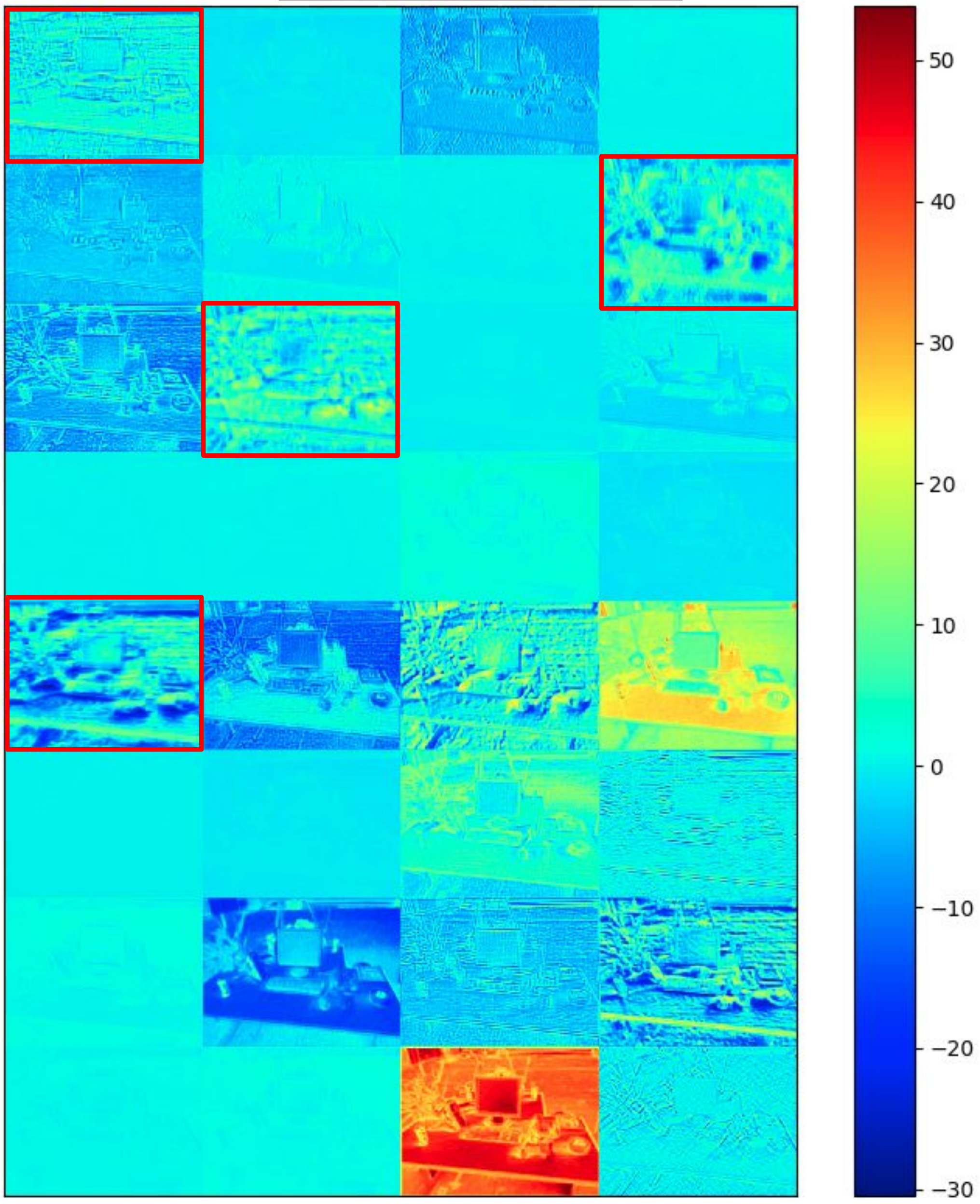


Figure 4.19: Output of the full version of our learned matching feature network (with all Blocks B1-B5). Interestingly most of the difference in the output of the fast and full versions of our network are concentrated in the feature maps highlighted by the red boxes.

Bibliography

S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems 30*. 2016.

J. Czarnowski, S. Leutenegger, and A. J. Davison. Semantic texture for robust dense tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 860–868, 2017.

D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

J. M. Fácil, A. Concha, L. Montesano, and J. Civera. Deep single and direct multi-view depth fusion. *CoRR*, abs/1611.07245, 2016. URL http://arxiv.org/abs/1611.07245.

R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

C. Godard, O. Mac Aodha, and G. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611. IEEE, 2017.

A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, May 2014.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2938–2946. IEEE, 2015.

A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 2017a.

A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. *CoRR*, abs/1703.04309, 2017b. URL http://arxiv.org/abs/1703.04309.

G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1925–1934, 2017.

C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. In *Dense Image Correspondences for Computer Vision*, pages 15–49. Springer, 2016.

B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981.

L. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238, 1989.

R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, abs/1610.06475, 2016. URL http://arxiv.org/abs/1610.06475.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

R. Newcombe. *Dense visual SLAM.* PhD thesis, Imperial College London, UK, 2012.

R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *IEEE ISMAR*. IEEE, 2011a.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011b.

V. Prisacariu, O. Kahler, M. Cheng, C. Ren, J. Valentin, P. Torr, I. Reid, and D. Murray. A Framework for the Volumetric Integration of Depth Images. *ArXiv e-prints*, 2014.

A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *CoRR*, abs/1611.00850, 2016. URL http://arxiv.org/abs/1611.00850.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

T. Schmidt, R. Newcombe, and D. Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017.

N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012.

K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6565–6574. IEEE, 2017.

B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. *CoRR*, abs/1612.02401, 2016. URL http://arxiv.org/abs/1612.02401.

C. Wang, H. K. Galoogahi, C. Lin, and S. Lucey. Deep-lk for efficient adaptive object tracking. *CoRR*, abs/1705.06839, 2017. URL http://arxiv.org/abs/1705.06839.

C. S. Weerasekera, Y. Latif, R. Garg, and I. Reid. Dense monocular reconstruction using surface normals. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2524–2531, May 2017. doi: 10.1109/ICRA.2017.7989293.

S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: learned invariant feature transform. *CoRR*, abs/1603.09114, 2016. URL http://arxiv.org/abs/1603.09114.

J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016.

H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. 2018.

Chapter 5

Just-in-Time Reconstruction: Inpainting Sparse Maps using Single View Depth Predictors as Priors

Previously we explored how surface normals predicted by a CNN from a single image can serve as useful priors for accurate dense monocular reconstruction. Nevertheless, a CNN can also learn to directly predict a depth map from a single image. These learned depth maps provide useful information about long and short range pairwise depth relationships that is complementary to surface normal maps. In this chapter, we propose a probabilistic scale-invariant depth prior based on learned depth information for a task we coin as "just-in-time reconstruction". The aim now is to efficiently produce an accurate dense reconstruction given a single image and some sparse depth information as "seed" geometry. Among several future directions, we conclude this chapter by introducing a unified model that combines our ideas discussed in this chapter with those presented in the earlier chapters.

5.1 Introduction

Simultaneous Localization and Mapping (SLAM) is a well studied problem and is a backbone of many visual robotics systems. State-of-the-art large scale SLAM systems like ORB-SLAM (Mur-Artal et al. [2015]) and LSD-SLAM (Engel et al. [2014]) successfully integrate information from multiple views to build a consistent map while also accounting for the map's uncertainty.

Most SLAM systems — for mostly practical reasons — use a sparse or semi-dense map. Although these sparse/semi-dense maps are very useful for tracking and localizing a robot in an office or even a autonomous vehicle in a city, a richer representation is often desired for successful navigation and for the robot to interact with the environment once it has located itself in the map. In contrast, systems like DTAM (Newcombe et al. [2011]) produce dense maps but storage is an issue.

In this work we advocate a more flexible approach where a denser representation of a small portion of the map can be rapidly generated from a sparse 3D point cloud *on-demand* by a task requiring richer understanding of the scene – we refer to this as "just-in-time reconstruction".

Image-guided inpainting using hand-crafted locally connected (Levin et al. [2004]; Petschnigg et al.

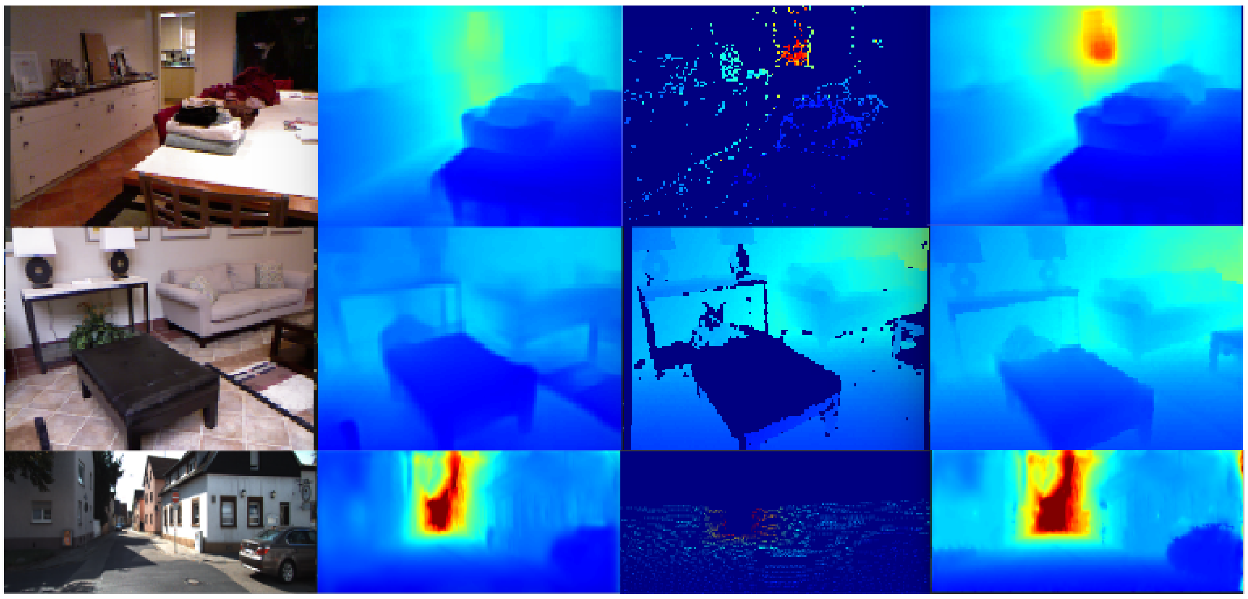


Figure 5.1: Three possible scenarios where our method can be applied. Top to Bottom: Image-guided inpainting of an ORB-SLAM depth map (with unknown scale, outliers, and irregular sparsity), a Kinect depth map (with structured holes), and a LIDAR depth map (with limited range and coverage) respectively. Left to Right: An input image, depth map predicted from input image, sparse/incomplete depth map from SLAM/sensor, our fused dense depth map.

[2004]) or fully connected Gaussian (Krähenbühl and Koltun [2011]) CRFs with learnable pairwise potentials (on depth and image colors) present themselves as obvious choices for the task. However, these CRFs ignore the arbitrary scale, varied irregular sparsity and/or uncertainty at which visual SLAM maps are created. Moreover they are very restrictive, relying on hand-crafted priors that link image colours and textures to depths (see Section 5.4.1).

On the other hand, in recent years, CNNs have emerged as immensely successful tool to directly learn the highly non-linear relationships between RGB images and depth in a more flexible way (Dharmasiri et al. [2017]; Eigen and Fergus [2015]; Kendall et al. [2017]; Laina et al. [2016]; Liu et al. [2016]; Ummenhofer et al. [2017]) through supervised, or even unsupervised (Garg et al. [2016]) means. While CNNs are not as easily scalable as most visual SLAM methods to incorporate information from hundreds of images at inference time, our view is that these depth prediction networks, in the form of millions of learnable weights, naturally allow for a much richer understanding about the distributions of valid partial maps conditioned on image patches over a large set of correlated images. Here we explore how to take advantage of this rich and flexible representation for "just-in-time reconstruction" to densify a sparse map that already incorporates complementary information from multi-view geometry or a depth sensor.

More formally, we present "just-in-time reconstruction" as a novel approach to real-time image-guided inpainting of a point-cloud obtained at an arbitrary scale and sparsity, to generate a fully dense depth map corresponding to the image. In particular, our goal is to inpaint a sparse map – obtained from either a monocular visual SLAM system or a sparse sensor – using a neural network which predicts dense depths from a single RGB image as a virtual sensor. In other words we want to probabilistically fuse a sparse SLAM map with information from a single-view depth prediction neural network.

This fusion process is however non-trivial due to structured errors present in single view depth predictions as well as the irregular sparsity and arbitrary scale of the SLAM map. For instance, depth maps from monocular visual SLAM are typically accurate at regions corresponding to high image gradient but correct only *up-to-scale*, and are usually irregularly sparse and can contain outliers. On the other hand depth predictions from a CNN are usually in metric scale, and dense, but have varying amount of errors depending on the capacity and generalization capability of the network used. Similarly depth maps from dedicated depth sensors, while being accurate, still suffer from the limited range and coverage of the sensors.

Adapting standard practices of probabilistic data fusion, we formulate just-in-time reconstruction as an inference problem over a novel learnable fully-connected Conditional Random Field (CRF). Nodes of this CRF penalize absolute depth error of the reconstructed depth map from the sparse observations stored in a large map while the edges enforce the pair-wise scale invariant depth relations (in this work, depth ratios of any two points in the image) of the inferred depth map to be the same as that of a single-view depth predictor while accounting for per-point confidence in the sparse map as well as confidence in the dense depth predictions.

Particularly, we obtain the confidence weights that parameterize the CRF model in a data-dependent manner via Convolutional Neural Networks (CNNs) which are trained to model the conditional depth error distributions given each source of input depth map and the associated RGB image. Section 5.2 outlines our novel CRF in detail which is (i) flexible enough to account for prediction uncertainties of sparse as well as dense maps, and (ii) allows for linear-time inference in real-time.

Our CRF formulation is very general and can be used to probabilistically fuse an arbitrary number of (sparse or dense) depth maps obtained from multiple input sensors. We also extend uncertainty estimation methods like that of Kendall and Gal [2017] to predict confidences for sparse and dense maps obtained at arbitrary scale (Sections 5.2.3 and 5.4.3) for use as parameters of our CRF.

We demonstrate effectiveness of the proposed "just-in time-reconstruction" approach by inpainting sparse maps with varied irregular sparsity and scale variations. In particular we show inpainting of (i) ORB-SLAM depth maps with scale ambiguity, outliers and highly irregular sparsity (ii) partially incomplete depth maps of indoor scenes obtained by a Kinect sensor and (iii) sparse point clouds obtained outdoors using a LIDAR sensor which have limited range and density, guided by single-view depth predictions and learned pointwise confidences. Figure 5.1 presents a snapshot of our results. Our just-in-time reconstruction approach generates depth maps with richer structural details than the semi-dense point clouds and gives more accurate reconstructions outperforming state-of-the-art image guided inpainting baselines and single-view depth prediction networks.

5.2 Just-in-Time Reconstruction

We define just-in-time reconstruction as an inference over a fully connected learnable CRF to inpaint a sparse map with P points that has been aligned with a single RGB image I with N pixels and represented as a *partial* log-depth map $y^s = [y_1^s, .., y_N^s] = ln(d^s) = [ln(d_1^s), .., ln(d_N^s)]$, where d_i^s is the depth of pixel i, with valid depths only at the projected pixel locations. Assuming that $c^s = [c_1^s, .., c_N^s], 0 \leq c_i^s \leq 1, \forall i$ is the confidence associated with the map (0 for invalid depths) estimated by a probabilistic SLAM approach or learned using a neural network trained to predict map confidence from input data (Section

5.2.3 and 5.4.1), our goal is to infer the dense log-depth map $y = [y_1, .., y_N] = ln(d) = [ln(d_1), .., ln(d_N)]$. We also assume that for this inpainting task we are given a dense depth map, regressed from a single view depth prediction network, and an associated confidence map (Sections 5.2.3 and 5.4.1). We denote the log-depths regressed by the depth prediction network to be $y^d = [y^d_1, .., y^d_N]$, and respective confidence maps to be $c^d = [c^d_1, .., c^d_N], 0 \leq c^d_i \leq 1$.

5.2.1 Model Definition

To achieve the "just-in-time reconstruction" from an image as described above we propose to minimize the following CRF energy w.r.t. y:

$$E(y) = \alpha E_u(y, y^s, c^s) + \beta E_{fc}(y, y^d, c^d) + \gamma E_{lc}(y, y^d, c^d) \tag{5.1}$$

where, E_u is the unary term generating the log depths for image I consistent with the sparse map, E_{fc} is a fully-connected pairwise term and E_{lc} is a locally connected pairwise term, both penalizing incorrect pairwise depth relationships (depth ratios as scale invariant measures) of the inferred dense log depth map using the single view depth predictions as the learned priors.[1] The tunable parameters $(\alpha, \beta, \gamma) > 0$ signify the relative importance of each term. A detailed description of each CRF term, the motivation behind using them and relations of these to existing frameworks are described in the following subsections.

Nodes of CRF

The unary term in the CRF pulls the inferred depth map to be consistent with the sparse map obtained via SLAM or a sensor. Similar to Eigen and Fergus [2015]; Eigen et al. [2014], we use squared log-depth differences for every point on the sparse map as the unary potentials of our just-in-time reconstruction CRF:

$$E_u(y) = \sum_i^N c^s_i (y_i - y^s_i)^2 \tag{5.2}$$

Notice that each term in E_u is weighted by the learned confidence of map accuracy c^s_i, and is accumulated over only the points present in the sparse map because $c^s_i = 0$ for pixels without a depth.

How does the log-depth parameterization help? The log depth parameterization makes the distribution of the residuals more homoscedastic (i.e. have more uniform variance throughout the solution space) (Ladický et al. [2014]). This happens because depth errors for far points (which are likely to be larger) are attenuated in log-depth parameterization. Homoscedasticity in the distribution is important as, according to the Gauss–Markov theorem, it is one the conditions for the ordinary least squares estimator to give the best linear unbiased estimate (Hallin [2014]; Odell [2014]).

The variance of errors in log-space ($ln(d)$) is similar to that in inverse depth space ($1/d$). We can prove this by considering the tangent line approximation of the respective errors with respect to depth (d) at a certain depth value a:

[1]In a general form multiple sparse and dense depth maps obtained at arbitrary scale from various sources can be used in our framework, replacing the CNN-based single view depth prediction, and simply minimizing the sum of the pairwise and fully connected terms described above, for each given depth map, in order to inpaint the desired sparse map.

$$\begin{aligned} Var(ln(d) - ln(d^s))_{d=a} &\approx Var(ln(a) + \frac{1}{a}(d-a) - ln(d^s)) \\ Var(ln(a) + \frac{1}{a}(d-a) - ln(d^s)) &= \frac{1}{a^2}Var(d) \end{aligned} \tag{5.3}$$

$$\begin{aligned} Var(\frac{1}{d} - \frac{1}{d^s})_{d=a} &\approx Var(\frac{1}{a} - \frac{1}{a^2}(d-a) - \frac{1}{d^s}) \\ Var(\frac{1}{a} - \frac{1}{a^2}(d-a) - \frac{1}{d^s}) &= \frac{1}{a^4}Var(d) \end{aligned} \tag{5.4}$$

We can see from the above equations that variance in depth is attenuated in log depth parameterization but not as much as in inverse depth parameterization (in Section 5.4.2 we propose an inverse-depth parameterization of our overall energy).

Edges of Fully Connected CRF model

As our goal is to inpaint sparse maps of arbitrary scale we aim to design a learnable prior which is insensitive to the scale of the scene. For this purpose, we propose the pairwise potentials of our full connected CRF to be:

$$E_{fc}(y) = \frac{1}{2N}\sum_{i,j} c^d_{ij}\big((y_j - y_i) - (y^d_j - y^d_i)\big)^2 \tag{5.5}$$

where c_{ij} are the learnable parameters of our CRF intuitively representing correctness of the log of depth-ratio $ln(d_j/d_i) = y^d_j - y^d_i$ of the single view depth predictions for any two points i and j to be learned in a data driven fashion.

Pairwise terms of our fully connected CRF can be best interpreted as the terms which enforce scale-invariant ordinal relationships of the inferred depths of two points to be same as that of the single view depth prediction network. The intuition behind using this fully connected CRF is that depth ratios of two points in a scene are invariant to the scale of the scene. Any other scale invariant function $f(d_i, d_j)$ may be used without loss of generality in our framework in place of $ln(d_j/d_i)$ (Section 5.4.2).

The fully connected pairwise CRF defined in (5.5) is however intractable in its most generic form, as the number of learnable parameters $c^d_{ij}, \forall(i,j)$ grows quadratically with the number of pixels in the image. Approximations are generally used to model c^d_{ij} in parametric form for reducing the number of independent learnable parameters and for efficient inference. The most common practice is to model c^d_{ij} as the sum of Gaussian RBF kernels each having two learnable parameters, which are mean and variance. For example Barron and Poole [2016]; Krähenbühl and Koltun [2011]; Levin et al. [2004]; Petschnigg et al. [2004]; Tomasi and Manduchi [1998] define c^d_{ij} in the form of Gaussian RBF kernels that are a function of the distance between pixel i and j and the color difference between those pixels. These CRF models allow for fast inference but are very restrictive.

In this work, we propose a different relaxation to $c^d_{ij} = c^d_i c^d_j$ which allows for efficient inference while having many more learnable parameters for expressiveness. The intuition is that the accuracy of the pairwise term is limited by the least confident depth value forming the ratio, and thus the overall confidence can be approximately expressed as a product of individual ones. This simple approximation significantly reduces the number of parameters to learn, and also allows for tractable inference (Section 5.2.2) as the fully connected term in Equation (5.5) (and thereby its gradient) can now be re-written in an equivalent alternative form that allows for linear time computation:

$$\begin{aligned} E_{fc}(y) = \frac{1}{N}\textstyle\sum_j^N c_j^d \displaystyle\sum_i^N c_i^d (y_i - y_i^d)^2 \\ - \frac{1}{N}\Big(\sum_i^N c_i^d (y_i - y_i^d)\Big)^2 \end{aligned} \tag{5.6}$$

Local Grid Connected Edges of CRF

Additionally, we define a grid connected term of the CRF to give more importance to the local structures learned by the single view depth predictor:

$$E_{lc}(y) = \sum_{i,k} c_i^d c_k^d \big((y_k - y_i) - (y_k^d - y_i^d)\big)^2 \tag{5.7}$$

where $k \in \{+u(i), +v(i)\}$ denotes pixel locations to the right of and below pixel i in the image plane. This enforces the solution to trust pairwise relations in y^d mainly around a local neighborhood around each pixel i, and is thus helpful for providing local support for the unary term where depth information is absent, where the solution is otherwise increasingly biased towards y^d as information from the dense fully connected terms dominates the weak information of y^s carried over from the unary potentials.

The end-effect of E_{lc} is similar to a data-driven local smoothing (Figure. 5.2) where information from y^d's local pairwise depth relationships are used to "smooth over" the areas where the sparse map's points are fused in the solution, while still anchoring the solution onto the sparse map. Apart from computational efficiency, we only consider a 4-connected graph for E_{lc} as E_{fc} already encompasses the full pairwise connectivity graph and, in addition to incorporating pixelwise confidences, our formulation provides the flexibility of using the tunable weights β and γ to control how much the solution gets biased towards y^s and y^d for a fixed α (Figure 5.4). We denote the set of pixels in the neighborhood of i as $\mathcal{N}(i)$. Note that for the grid-connected terms, the approximation $c_{ik}^d = c_i^d c_k^d$ does not offer as much computational benefit as before as only 2N (N horizontal and N vertical direction) confidences need to be learned and regressed otherwise (as opposed to $N(N-1)/2$ in the fully connected case).

Additional model expressibility can be added to our locally connected pairwise terms by increasing the size of $\mathcal{N}(i)$, and for instance introducing a multiplicative pairwise pixel-distance based Gaussian RBF kernel with tunable variance to the terms in Equation (5.7) — such that nearby pairwise depth ratio inconsistencies are penalized more strongly than those further apart — while still retaining the same inference method. We suggest potential extensions to our model in Section 5.4.1.

5.2.2 Inference Method

The inference objective is to find $min_y E(y)$. For ease of expression we can re-write (5.1) in the following form:

$$\begin{aligned} E(y) = y^T A y - 2(y^T A^s y^s + y^T A^d y^d) \\ + y^{sT} A^s y^s + y^{dT} A^d y^d \end{aligned} \tag{5.8}$$

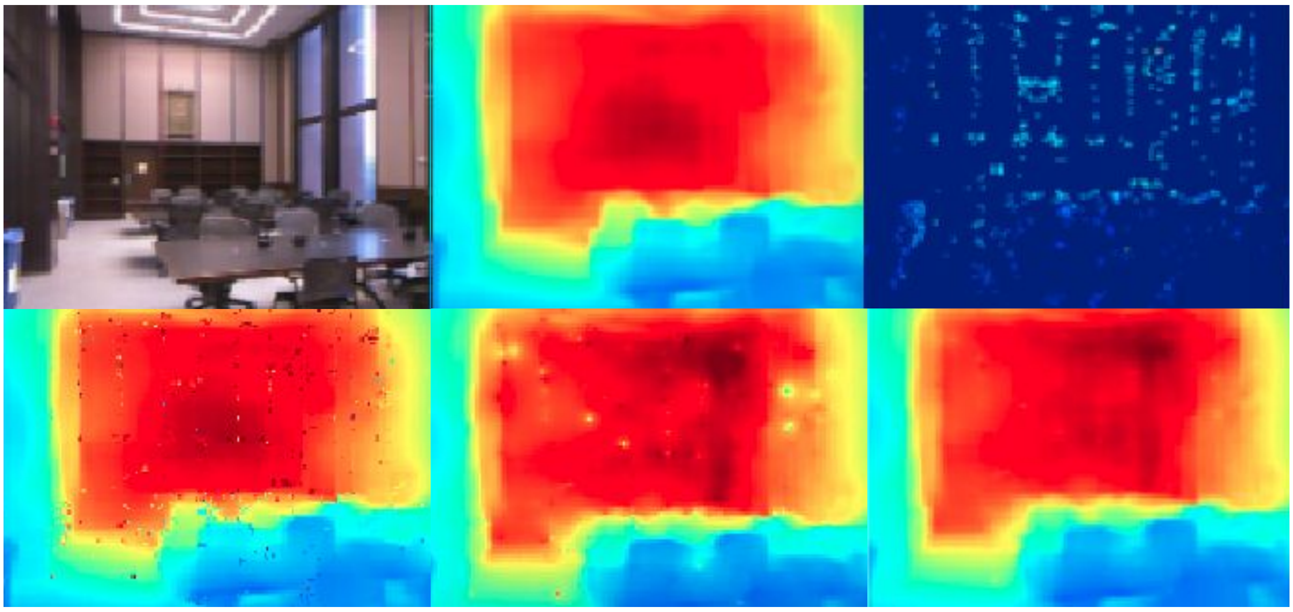


Figure 5.2: Qualitative ablation of the impact of local "data-driven smoothing" on the reconstruction as enforced by our local CRF pairwise terms E_{lc}, and the impact of incorporating confidences into the energy. In the first row from left: (1) RGB Image, (2) Dense depth prediction of Eigen and Fergus [2015] (metric scale), (3) Sparse depth map of Mur-Artal et al. [2015] (unknown scale). In the second row from left: (4) Our reconstruction with learned confidences but without E_{lc}, (5) Our reconstruction with E_{lc} but without learned confidences, (6) Our final reconstruction with both E_{lc} and learned confidences. Our E_{lc} (local "data-driven smoothing") terms enforce short range pairwise depth relationships extracted from the learned depth map in the reconstruction. They compensate for inaccuracies in long range pairwise depth relationships in the learned depth map that are enforced by our fully connected pairwise terms E_{fc}. The learned depth confidences help further suppress inaccuracies and outliers in both the learned depth map and ORB-SLAM depth map.

where $A = (A^s + A^d)$ is a $N \times N$ symmetric and positive (semi-) definite matrix. A^s is a diagonal matrix with entries $A^s_{ii} = \alpha c^s_i, \forall i$ while A^d is a dense $N \times N$ symmetric and positive (semi-) definite matrix with entries as follows:

$$\begin{aligned} A^d_{ii} &= c^d_i \Big(\frac{\beta}{N} \textstyle\sum_{j,j\neq i}^{N} c^d_j + \gamma \sum_{j\in\mathcal{N}(i)} c^d_j \Big) && \forall i \\ A^d_{ij} &= -c^d_i \Big(\frac{\beta}{N} c^d_j + \gamma c^d_j \Big) && \forall i, j \in \mathcal{N}(i) \\ A^d_{ij} &= -c^d_i \Big(\frac{\beta}{N} c^d_j \Big) && \forall i, j \neq i, j \notin \mathcal{N}(i) \end{aligned} \tag{5.9}$$

Differentiating (5.8) with respect to y and then setting the resulting expression to 0 we obtain:

$$Ay = A^s y^s + A^d y^d \tag{5.10}$$

To solve for y in (5.10) we use the iterative conjugate gradient method (Algorithm 5.1). We could also solve for y in closed form but this would be less efficient, especially when N is large. For the algorithm we do not need to explicitly construct matrices A^s and A^d but simply evaluate the gradients $A^s y^s$ and $A^d y^d$ at the start, and Ap_k at each iteration-step. Note that computing $A^d y^d$ and Ap_k require

Algorithm 5.1: Iterative conjugate gradient method for solving for the optimal log depth map $\mathbf{y}$ for an image $\mathbf{I}$. Note that vectors here are denoted in bold unlike in the text for clarity.

```
1  Initialize $\mathbf{y}_0 = \mathbf{0}$ ;
2  Initialize $\mathbf{r}_0 = A^s\mathbf{y}^\mathbf{s} + A^d\mathbf{y}^\mathbf{d} - A\mathbf{y}_0$ ;
3  Initialize $r_0 = \mathbf{r}_0^T\mathbf{r}_0$ ;
4  Initialize $\mathbf{p}_0 = \mathbf{r}_0$ ;
5  Initialize $k = 0$ ;
6  repeat
7      $\alpha_k = \frac{r_k}{\mathbf{p}_k^T A\mathbf{p}_k}$ ;
8      $\mathbf{y}_{k+1} = \mathbf{y}_k + \alpha_k\mathbf{p}_k$ ;
9      $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$ ;
10     $r_{k+1} = \mathbf{r}_{k+1}^T\mathbf{r}_{k+1}$ ;
11     if $r_{k+1} < \epsilon$ exit loop ;
12     $\beta_k = \frac{r_{k+1}}{r_k}$ ;
13     $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$ ;
14     $k = k + 1$
15 until convergence;
16 $\mathbf{y} = \mathbf{y}_{k+1}$
```

$O(N^2)$ operations, however based on the simplified form of equation (5.6), we could compute them in linear time by evaluating:

$$(Ap_k)_i = \alpha c_i^s(p_k)_i + c_i^d\Big(\frac{\beta}{N}\sum_j^N c_j^d(p_k)_i - \frac{\beta}{N}\sum_j^N c_j^d(p_k)_j + \gamma\sum_{j\in\mathcal{N}(i)} c_j^d((p_k)_i - (p_k)_j)\Big), \forall i \tag{5.11}$$

at each time step k, and similarly:

$$(A^d y^d)_i = c_i^d\Big(\frac{\beta}{N}\sum_j^N c_j^d y_i^d - \frac{\beta}{N}\sum_j^N c_j^d y_j^d + \gamma\sum_{j\in\mathcal{N}(i)} c_j^d(y_i^d - y_j^d)\Big), \forall i \tag{5.12}$$

once at the start. Both (5.11) and (5.12) are linear time operations as the terms $\sum_j^N c_j^d$, $\sum_j^N c_j^d(p_k)_j$ and $\sum_j^N c_j^d y_j^d$ are common $\forall i$ and can be re-used.

In practice the solution converges rapidly to within a desired threshold in $n \ll N$ iterations. To further accelerate the process we implement the algorithm to run on the GPU, where per-pixel operations such as computation of unary and local pairwise terms are parallelized, and sum operations for the fully connected term are accelerated via parallel reduction.

Note that for the system of linear equations to have a unique solution, i.e. non-zero determinant for A, then α should be non-zero. Intuitively it means that at least one input depth map must contribute to the absolute scale of the fused depth map, else infinite solutions exist where the fused depth map is correct up-to-scale. Also, to prevent a potential condition number of infinity due to diagonal entries in A being equal to 0, we add a small ϵ to $c_i^d, \forall i$.

5.2.3 Learning to Predict Confidence Weights

The goal here is to learn separate CNN models that can model the conditional error distributions of y^s and y^d and predict confidence maps c^s and c^d respectively, given the respective depth maps and I as input. The motivation behind learning these confidence maps from the data is provided at the end of this section.

Since the training setup and network architecture is almost identical for the two cases, for brevity we focus on the training procedure for predicting c^s, and mention the differences. The training loss L^s for predicting c^s is defined as follows:

$$L^s = \frac{1}{|b|}\sum_{i}^{N} b_i^s(\hat{c}_i^s - c_i^s)^2 \tag{5.13}$$

where $\hat{c}^s$ is the predicted confidence map for y^s, b_i^s is a binary value which is 0 if *either* y_i^s or y_i (groundtruth) is invalid. Since confidence of a depth value is inversely proportional to its error (and in the more general case scale-invariant error) we define the groundtruth confidence map c_i^s as follows:

$$c_i^s = b_i^s e^{-\lambda^s |E_i^s|} \tag{5.14}$$

where $\lambda^s > 0$ is a tunable parameter controlling the contrast of c^s, and E_i^s is the scale-invariant error for a depth value in pixel i defined as:

$$\begin{aligned} E_i^s = {} & (\alpha^s + \beta^s) b_i^s (y_i^s - y_i) \\ & - \frac{\beta^s}{|b|} \sum_{j}^{N} b_j^s (y_j^s - y_j) \\ & + \gamma^s \sum_{j \in \mathcal{N}(i)} b_i^s b_j^s \big((y_j^s - y_i^s) - (y_j - y_i)\big) \end{aligned} \tag{5.15}$$

where y here stands for the groundtruth log-depth map. The parameters $(\alpha^s, \beta^s) > 0$ can be set based on how scale-invariant we require E_i^s to be ($\alpha^s = 0$ for full-scale-invariance in the case of y^s as it is in a random scale), and $\gamma^s > 0$ determines whether the network should emphasize more on learning confidences in local pairwise connectivity.

Note that in Section 5.4.3, as a future direction, we discuss an alternative loss function that will potentially enable us to learn a more accurate c^s and c^d to suit our model, in a more implicit manner.

The inputs to the CNN model are y^s and I. The RGB image is first passed through a (9x9 kernel size) convolutional layer with 127 output feature maps. The output feature maps are then concatenated with the input log-depth map and passed through 6 more (5x5 kernel size) convolution layers with each having 128 output feature maps, except for the last layer which regresses the confidence map $\hat{c}^s$. All layers are followed by ReLU activation functions, except the output layer which we leave as linear. At test time the predicted values are clipped between 0 and 1 inclusive.

The irregular sparsity structure in y^s at the network input poses a difficulty to the learning process, as the network has an additional task of learning which points in the input depth map are valid. This demands higher model capacity. One way to facilitate the learning process is to explicitly model the

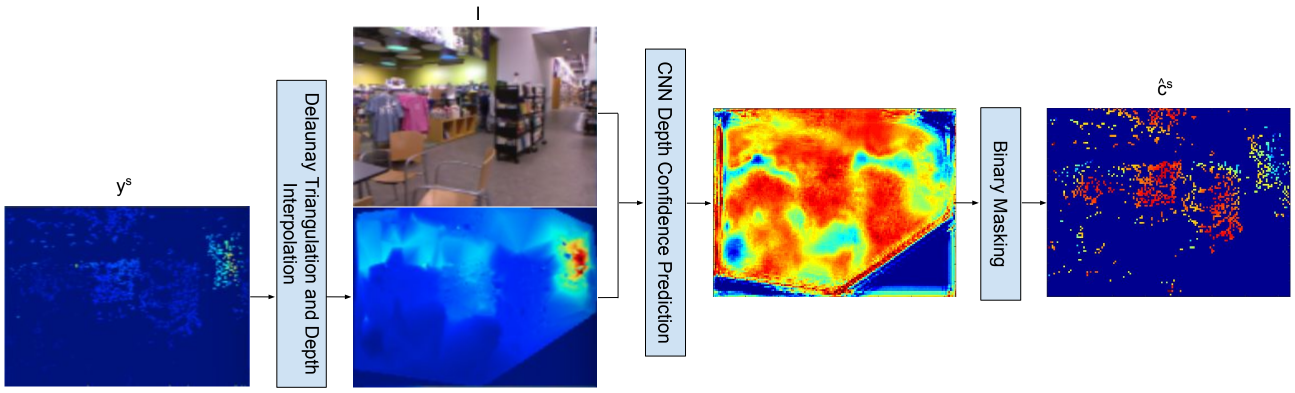

I
ys
Delaunay Triangulation and Depth Interpolation
CNN Depth Confidence Prediction
Binary Masking
ĉs

Figure 5.3: Figure shows the result of densification of a sparse map via Delaunay triangulation, for use as input to our confidence prediction CNN (along with the corresponding RGB image) and the final point-wise confidence prediction for the sparse map. Regions outside the convex hull can either be set to 0 or the mean depth (we chose the latter option). The red regions in the predicted confidence map denotes high confidence.

network to be invariant to the sparsity of the data, for instance by performing masked convolutions at each layer (Liu et al. [2018]; Uhrig et al. [2017]) which require significant computation overhead.

We believe that a more efficient way to facilitate learning with a small network is to densify the data itself before feeding it into the convolutional layers based on some assumption about the data. Here we opted to perform Delaunay triangulation on the 2D image coordinates corresponding to valid points on the depth map, followed by barycentric-coordinate-based linear interpolation (in inverse depth space) to fill the triangles with depth values. The latter can be efficiently carried out on the GPU. This densification method is motivated by the fact that most regions in a depth map are typically piecewise-planar. Doing so also enhanced errors in the sparse depth map that otherwise would have been difficult for the network to pick up. We also found that using the triangulated dense log depth map in place of y_i^s in equation (5.15) gave more reliable estimates of E_i^s. Figure 5.3 shows the result of the Delaunay triangulation.

Note that future work can explore alternative representations of sparse depth maps, that can serve as input to a CNN predicting depth confidence. Even though computationally more expensive, it is also worth exploring and improving upon CNNs like Liu et al. [2018]; Uhrig et al. [2017] that can directly deal with sparse data at the input. Furthermore, future work can experiment with predicting the confidences c^s and c^d from the input image I alone, thereby not needing to deal with the issue of the sparsity of y^s at the input to the network.

In order to get $\hat{c}^s$ from the network output, i.e. the densely predicted confidence map corresponding to y^s, we simply perform an element-wise multiplication of the network output with y^s's binary mask (as illustrated by the example in Figure 5.3). The random variability in map scale of y^s also poses a difficulty to the learning process, and as a solution to this we scale each triangulated dense log depth map so that its mean is equal to the mean of the groundtruth log depths in the entire train set, before passing it into the network. For training, we use the Caffe (Jia et al. [2014]) framework. Training was performed with a batch size of 16, a learning rate of $1e^{-2}$, momentum of 0.9 using SGD as the optimizer on a NVIDIA GTX 1080 Ti GPU.

How do the confidence weights help? The confidence weights (which ideally are proportional to the inverse-variance of the residuals) model the heteroscedastic nature of the error distribution caused by outliers or other structured errors in observed depths. The more accurate modeling of the multi-variate Gaussian distribution favors the ordinary least squares estimator when inferring the solution y. Note that one can choose to use the L1 norm of the residuals which assumes a Laplace distribution, and obtain the best median estimate rather than best mean estimate, as the former is more robust to heteroscedasticity. This is infact feasible for our unary terms.

However applying L1 norm on the pairwise terms of our fully connected energy makes inference computationally expensive if for instance we were to use the Iteratively Re-Weighted Least Squares (IRLS) method. This is because at each step of the IRLS algorithm we need to compute the associated weighting for each term (proportional to the inverse of the term), and this step would be very expensive to perform on all the edges of a fully-connected CRF. By directly predicting the confidence weights from the data, we can simply apply the ordinary least squares estimator. Note that it is also highly inefficient to apply methods like the primal-dual algorithm in a fully connected CRF due to the explosion in the number of dual variables required.

Certain SLAM methods maintain a confidence weight for the map internally (Engel et al. [2014]; Mur-Artal et al. [2015]) based on multi-view color/feature consistency. However, specular surfaces for instance may cause points to be incorrectly reconstructed as being far away, and the typical SLAM algorithm is unaware of such cases as these incorrectly reconstructed landmarks still pass the multi-view consistency check. The CNN confidence predictor, while also prone to error, can successfully learn to predict a low confidence weight for such outliers.

As a further step to minimize the amount of outliers in the projected sparse depth map, and also account for changes in the environment, the features corresponding to each 3D point can be compared against those of the input RGB image at the corresponding projections. If the feature error is greater than a threshold we can discard these points in the depth map assuming they are occluded or the environment has changed.

5.3 Experimental Results

This section provides the quantitative and qualitative results of our approach for different experimental settings on NYU Depth v2 (Silberman et al. [2012]) and KITTI (Geiger et al. [2013]) datasets, where we perform image-guided inpainting of sparse/semi-dense ORB-SLAM, Kinect, and LIDAR depth maps. We use single view depth predictions from Eigen and Fergus [2015] and Garg et al. [2016] for the NYU and KITTI datasets respectively. We also compare against two other image-guided depth inpainting methods as our baselines: the cross-bilateral filter (Petschnigg et al. [2004]), and colorization (Levin et al. [2004]). These inpainting methods are commonly used for Kinect depth inpainting in the NYU dataset and we directly use the code available from the NYU toolbox (Silberman et al. [2012]).

5.3.1 Inpainting Sparse Depth Maps from Multi-View Geometry

We first demonstrate just-in-time reconstructions of sparse ORB-SLAM maps that are projected onto frames in the subset of the NYU dataset. For this experiment, we use the train/test split specified in Eigen and Fergus [2015], however our train/test set is a fraction of theirs limited by the success

Table 5.1: Quantitative results of inpainting ORB-SLAM maps on NYUv2 dataset, followed by an ablation of the effects of incorporating learned pointwise confidences of the sparse map and predicted depth map.

Method	Scale Invariant Error
Sparse Map (Mur-Artal et al. [2015])	0.492
Depth Map Prediction (Eigen and Fergus [2015])	0.159
Cross-bilateral Filter	0.491
Colorization	0.372
Ours	**0.144**

Learned Confidence		Scale Invariant Error
Map	Prediction	
x	x	0.150
x	✓	0.149
✓	x	0.145
✓	✓	**0.144**

of ORB-SLAM's (Mur-Artal et al. [2015]) tracking on the corresponding scenes in the raw dataset. We use the network proposed in Eigen and Fergus [2015] as the virtual depth sensor for all indoor experiments.

Quantitative results of our just-in-time reconstruction are summarized in Table 5.1 where we report the average scale invariant error measure (sc-inv), which was also used as a performance measure in Ummenhofer et al. [2017]:

$$\text{sc-inv} = \sqrt{\frac{1}{M}\sum_{m}^{M}\frac{1}{|b^{(m)}|}\sum_{i}^{N} b_i^{(m)}(y_i^{(m)} - \hat{y}_i^{(m)})^2 - \frac{1}{|b^{(m)}|^2}\Big(\sum_{i}^{N} b_i^{(m)}(y_i^{(m)} - \hat{y}_i^{(m)})\Big)^2} \tag{5.16}$$

where $y^{(m)}$ and $\hat{y}^{(m)}$ are the groundtruth and estimated natural log depth maps respectively, for the m^{th} instance in the test set, and $b^{(m)}$ is the corresponding binary mask with zeros where depth is invalid in *either* $y^{(m)}$ or $\hat{y}^{(m)}$ at the corresponding pixel i.

Figure 5.4 shows the sensitivity of the just-in-time reconstruction as we vary the strength of the fully connected and the local grid connected terms of the CRF. It is evident that both the terms contribute to the performance of our ORB-SLAM depth inpainting. Table 5.1 shows that the CRF based inpainting results successfully take advantage of the accumulated geometric evidence from multiple-views contained in the sparse map in order to significantly improve upon the depth prediction errors of the neural network baseline. It is important to note that the ORB-SLAM sparse map scale invariant error measure is computed only on the points visible in the map so does not precisely correspond to that of denser error measures but still indicative that ORB-SLAM maps contain gross outliers. The table shows that we also significantly outperform the two other image-guided inpainting baselines: the cross-bilateral filter (Petschnigg et al. [2004]), and colorization (Levin et al. [2004]). Both of these methods use relatively simpler models that assume depth within a local region can be represented as a linear function of image color (see Section 5.4.1), and hence they produce unrealistic depth maps most of the time, especially when the amount of sparsity is high.

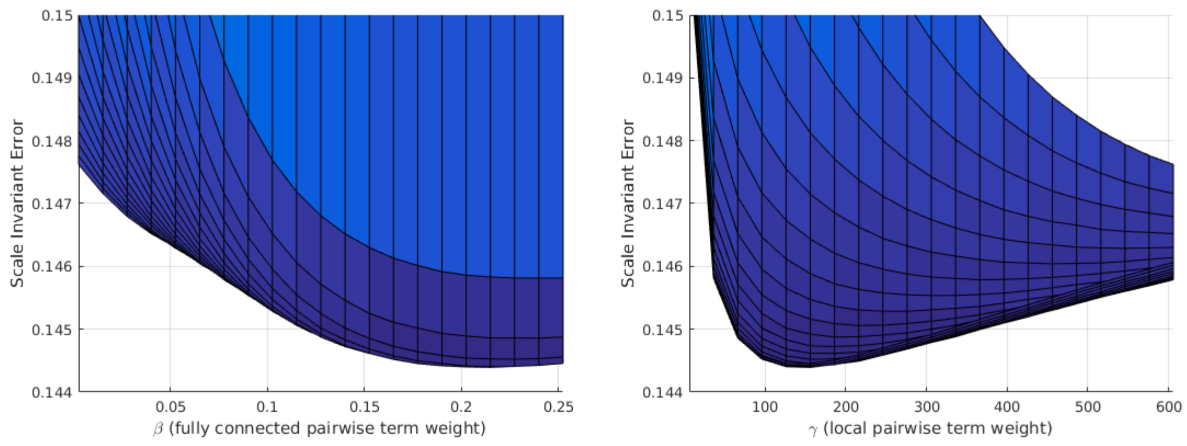


Figure 5.4: Plots showing the average scale invariant error (on vertical axis) of our just-in-time reconstructions for sparse ORB-SLAM map input on the NYU dataset as we vary the hyperparameters (on horizontal axis): β (left) and γ of our CRF, with $\alpha = 10$. The bottom-most curve in each plot represents the best scale-invariant reconstruction error against the varied fully connected term strength (β) and locally connected term strength (γ) respectively.

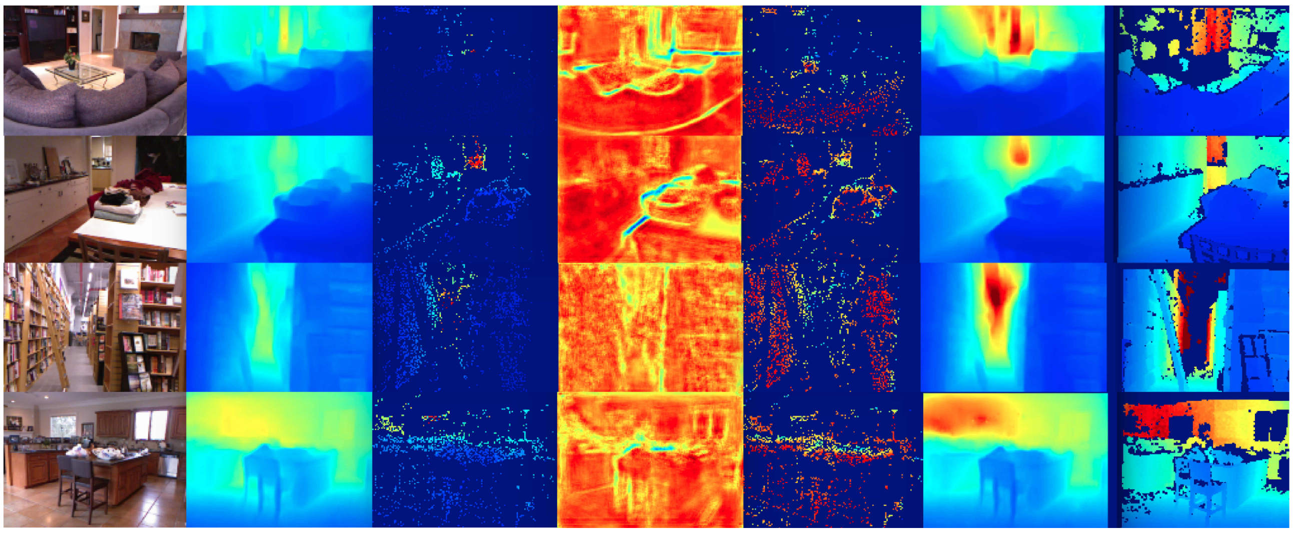


Figure 5.5: ORB-SLAM inpainting results for some of the images in the NYU test set. In column order: (1) RGB image, (2) Depth map prediction (Eigen and Fergus [2015]), (3) ORB-SLAM depth map, (4) Predicted confidences using our method for depth predictions (red implies higher confidence), (5) Predicted confidence using our method for the sparse map, (6) Our just-in-time reconstruction result, (7) Ground truth.

Figure 5.5 shows the qualitative results of our just-in-time reconstruction approach for ORB-SLAM depth inpainting. Depth map predictions of Eigen and Fergus [2015] are often inaccurate at the edges – a likely artifact of loss in resolution from the single view depth estimation network – and at regions which are further away from the camera. Our confidence prediction networks most often correctly predicts the likely confidences of the input depth data (even for the sparse ORB-SLAM depth maps which are at arbitrary scale) which help reduce most of the depth errors and gross outliers from making it into the fused result. Figure 5.6 shows a visual comparison of how our approach performs against the two other image-guided depth inpainting baselines.

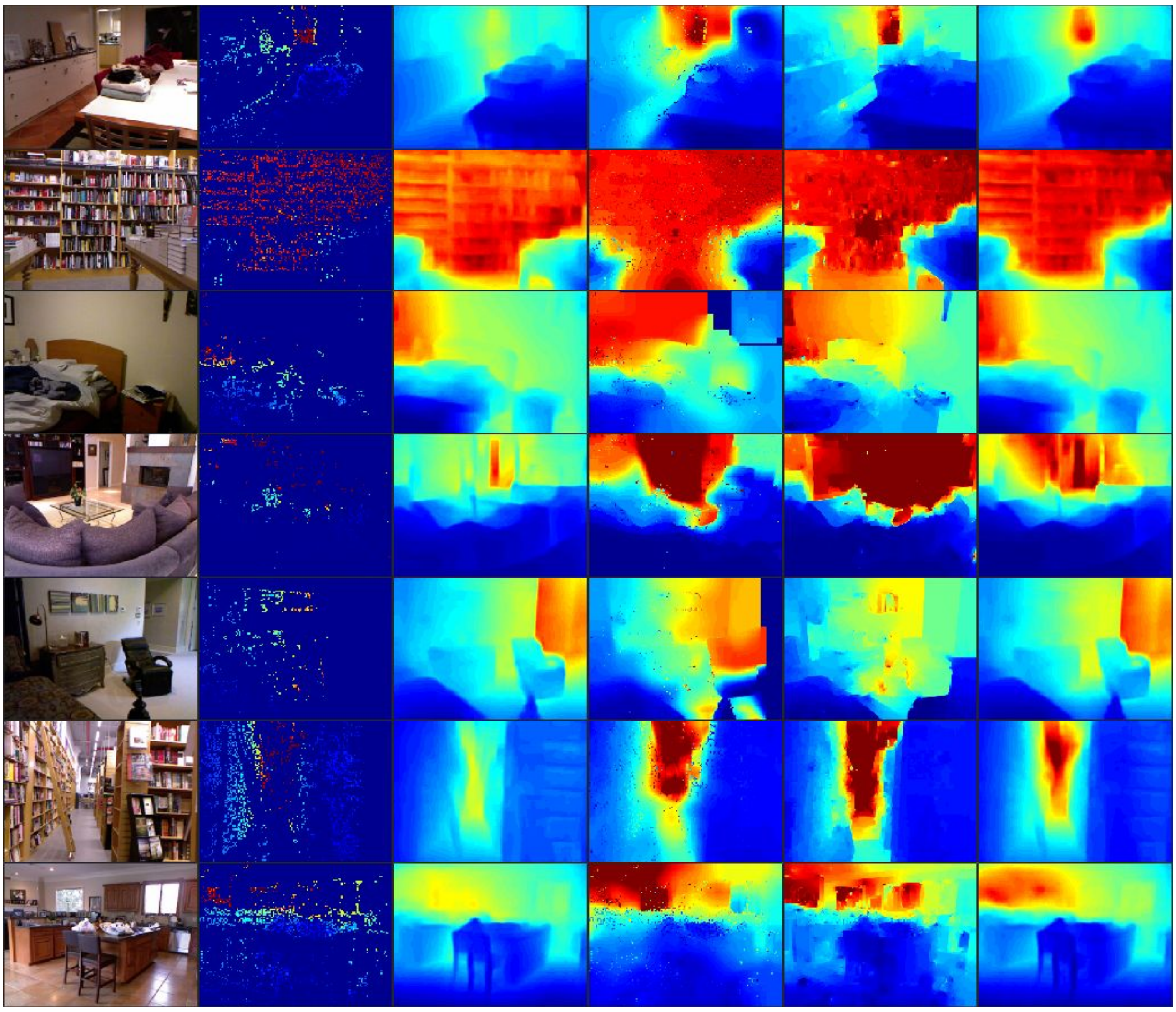


Figure 5.6: Comparison of our method against baselines for inpainting ORB-SLAM depth maps in the NYU dataset. In column order: (1) RGB image, (2) ORB-SLAM depth map, (3) Depth map prediction (Eigen and Fergus [2015]), (4) Cross-bilateral filter, (5) Colorization, (6) Our just-in-time reconstruction result.

In Table 5.1, we provide an ablation study justifying importance of confidence estimation. In the bottom part, we show that incorporating confidences of both the map and the single view depth predictor, is important for obtaining more accurate reconstructions. Dropping confidences estimated in either sparse map or dense depth predictor (or both), as expected, degrades the quality of the just-in-time reconstruction. Figure 5.2 shows the same visually.

Note that interestingly, if we want the final fused result to be closer to metric scale (the scale of the predicted depth map y^d) — while still having *similar* scale-invariant-error as the *up-to-scale* reconstruction obtained in our current setup (which is in the scale of the ORB-SLAM map) — we can simply use y^d in the unary terms, and y^s in the fully-connected term of our energy formulation (instead of the other way around), and re-tune the hyper-parameters α, β, and γ.

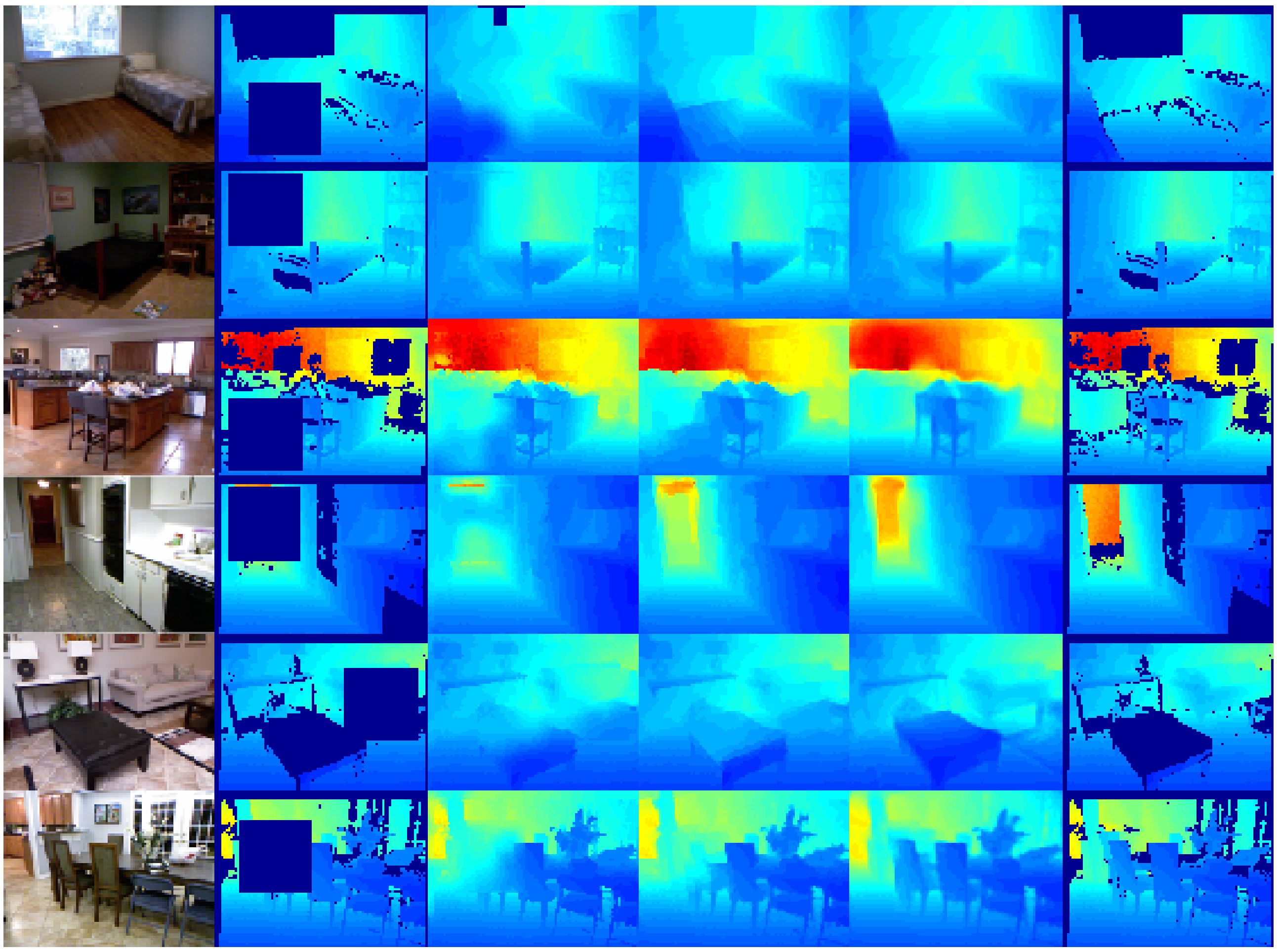


Figure 5.7: Comparison of our method against baselines for inpainting Kinect depth maps with randomly removed data on the NYU dataset. In column order: (1) RGB image, (2) Kinect depth map with randomly removed data, (3) Inpainted depth map using cross-bilateral filtering, (4) Inpainted depth map using Colorization, (5) Our just-in-time reconstruction result, (6) Raw Kinect depth map.

5.3.2 Inpainting Sparse Depth Maps from Depth Sensor

We also evaluate our method for inpainting sparse sensory data captured in indoor and outdoor environments. Note that in our experiments for sensory data inpainting, relying purely on our unary and local pairwise terms was sufficient. This is because unlike ORB-SLAM maps, the sensory data used here is fairly accurate with little to no outliers, and thus don't require a fully-connected pairwise neighbourhood of learned depth relationships to rectify the existing sensor data. Hence, we can fill-in the holes in the sensory depth maps based on the very local depth relationships of the single view depth predictions (only the local pairwise weight γ need to be tuned on a validation set for a fixed α). It is sensible to assume that the fully-connected term would be more beneficial if the sensor is noisy and unreliable, and/or if the depth map prediction (or whichever input depth source, whether sparse or dense, that will be used for inpainting) is more reliable, such as in the case for ORB-SLAM inpainting.

We first evaluate our method for inpainting Kinect depth maps on our NYU test subset, again using the network in Eigen and Fergus [2015] as the virtual depth sensor. To quantitatively evaluate the results, while respecting the structured sparsity pattern, we introduce a synthetic version of the

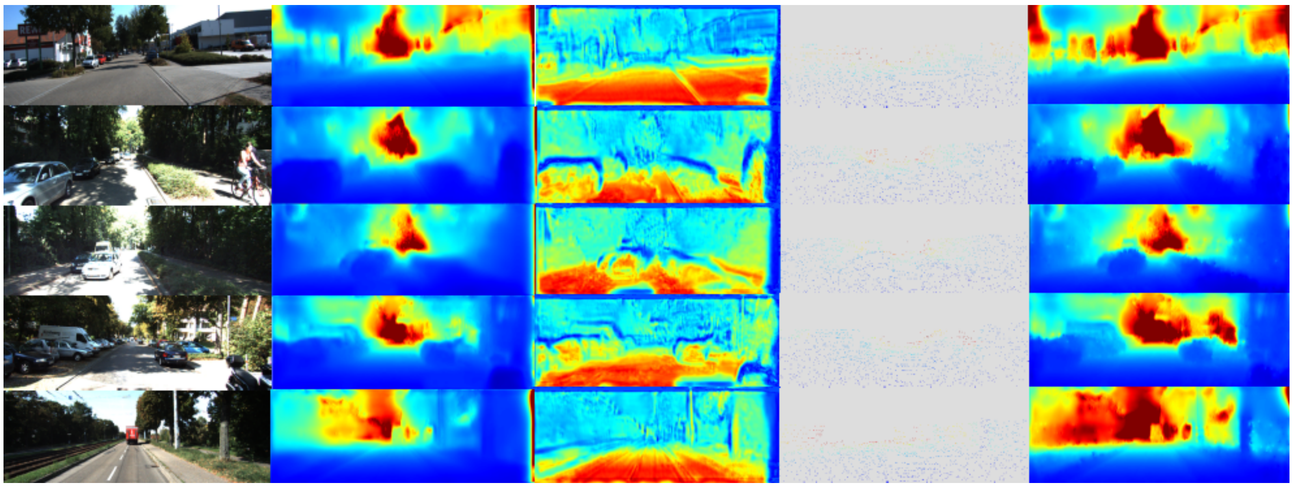


Figure 5.8: Intermediate and final outputs for LIDAR depth map inpainting in the KITTI dataset. For the quantitative evaluation in Table 5.3 we have randomly removed $2/3^{rd}$ of points from the maps making them even sparser. Notice that in spite of this we are able to densify the maps whilst being consistent with their values and significantly improving over the single view depth predictions. In column order: (1) RGB image, (2) Depth map prediction (Garg et al. [2016]), (3) Predicted confidences using our method for depth predictions (red denotes higher confidence), (4) LIDAR depth map with $2/3^{rd}$ of points randomly removed, (5) Our just-in-time reconstruction result.

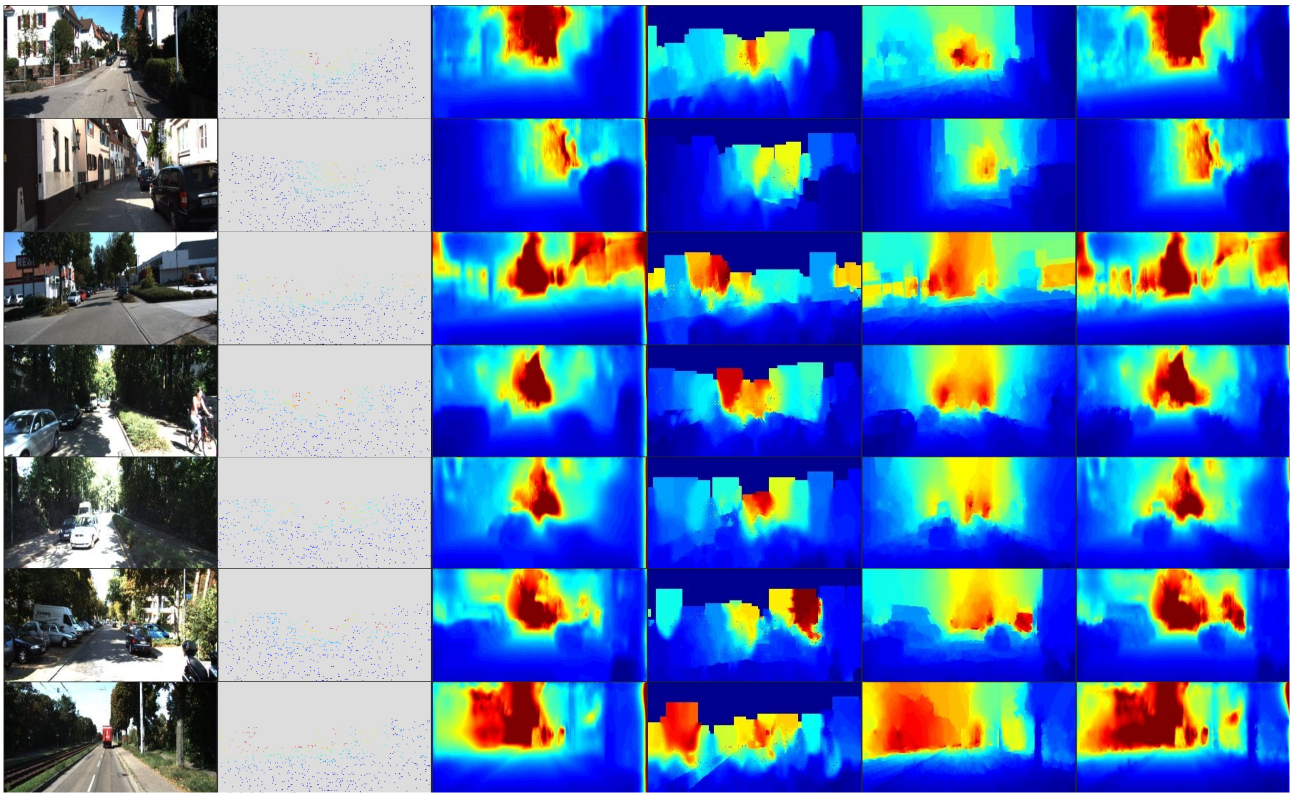


Figure 5.9: Comparison of our method against baselines for inpainting LIDAR depth maps in the KITTI dataset. In column order: (1) RGB image, (2) LIDAR depth map with $2/3^{rd}$ of points randomly removed, (3) Depth map prediction (Garg et al. [2016]), (4) Cross-bilateral filter, (5) Colorization, (6) Our just-in-time reconstruction result.

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen and Fergus [2015]	0.679	0.217	0.155	0.117	0.740	0.952	0.990
Cross-bilateral filter	0.249	0.077	0.023	0.019	0.975	0.993	0.998
Colorization	0.200	0.059	0.019	0.011	0.984	0.997	0.999
Ours	**0.169**	**0.047**	**0.018**	**0.007**	**0.991**	**0.999**	**1.000**

Table 5.2: Inpainting results on the NYU dataset for Kinect depth maps which are further sparsified by removing random crops. The results are evaluated against the original (downsampled) Kinect depth maps, on a *subset* of Eigen and Fergus [2015]'s test split (the same subset used for the ORB-SLAM inpainting experiments). Note that it is a common practice in the literature to evaluate against the original depth resolution of 640x480. However we performed inpainting at the resolution of 147x109 to increase efficiency and for real-time performance, and therefore did our evaluation against 147x109 resolution Kinect depth maps, downsampled using the nearest neighbour method. The random crops which are of size 50x50 were removed from the downsampled Kinect depth maps.

	Error (lower is better)				Accuracy (higher is better)		
	rms (m)	log	abs.rel	sq.rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Garg et al. [2016]	5.555	0.285	0.180	1.119	0.692	0.903	0.965
Cross-bilateral filter	3.854	0.195	0.113	0.764	0.853	0.955	0.983
Colorization	1.956	0.104	0.048	0.211	0.965	0.989	0.995
Ours	**1.838**	**0.092**	**0.032**	**0.146**	**0.969**	**0.990**	**0.996**

Table 5.3: Inpainting results on the KITTI dataset for LIDAR depth maps which are further sparsified by randomly removing a 2/3rd of its points. The results are evaluated against the original (downsampled) sparse LIDAR depth maps, in a subset of Eigen et al. [2014]'s test split. Note that it is a common practice in the literature to evaluate against the original depth resolution of 1242x375 within a masked pixel region of [44:1196, 153:370] (with 0-based indexing). However we performed inpainting at the resolution of 608x160 to increase efficiency and for real-time performance, and therefore did our evaluation against 608x160 resolution LIDAR depth maps, downsampled using the nearest neighbour method, within a masked pixel region of [21:585, 65:157].

NYU test set where a random rectangular region is cropped from the Kinect depth map to be labeled missing and then inpainted using the rest of the visible depth map. In Table 5.2, we report the inpainting accuracy on the original depth maps, and as before, compare with depth estimations from Eigen and Fergus [2015], and inpainting results using cross-bilateral filtering (Petschnigg et al. [2004]) and Colorization (Levin et al. [2004]) as baselines. Figure 5.7 shows the qualitative comparison of the results of our CRF based inpainting with that of the baselines. It is clear that the proposed method produces more realistic depth maps in comparison to the mostly piecewise constant and inaccurate depth maps that the two other low-level image-guided depth in painting baselines produce.

Next we evaluate our method for inpainting sparse LIDAR maps in the KITTI dataset. To facilitate quantitative evaluation we remove $2/3^{rd}$ of the map points (respecting the sparsity structure of the LIDAR data) and evaluate the inpainted results against the original depth maps. Quantitative results on the test set are shown in Table 5.3 where our inpainting method improves greatly over the baseline CNN predictions of Garg et al. [2016], and the two image-guided depth inpainting baselines. It is interesting to note here that the colorization method performs quite well in spite of its simpler model, and we provide a suggestion in Section 5.4.1 as to how we can draw inspiration from this model to potentially further improve our results. Some qualitative intermediate and final results of our method are shown in Figure 5.8. We can observe sharper object boundaries in our final inpainted result, in comparison to the blurred object boundaries in the predicted depth map due to the loss in resolution that very deep feed forward CNNs suffer from. Most other errors in scene structure in the depth predictions have also been corrected in our fused result. A qualitative comparison against all three baselines is provided in Figure 5.9.

5.3.3 Runtimes

In our test setup which uses a NVIDIA GTX 980 GPU and Intel i7 4790 CPU, inference time for ORB-SLAM and Kinect depth map inpainting on the NYU dataset is $\approx 30ms$ at 147x109 image resolution, which is the same resolution as the predicted depth map by Eigen and Fergus [2015]. Inference time is $\approx 200ms$ if performed at the full image resolution of 640x480 by upsampling depth predictions. Inference time for LIDAR depth map inpainting on the KITTI dataset is $\approx 100ms$ at 608x160 image resolution, which is the same resolution as the predicted depth map by Garg et al. [2016]. Total overhead time for CNN depth and confidence predictions is $\approx 50ms$ for both datasets.

5.4 Future Directions

5.4.1 Increasing Model Expressibility

Our relaxation of the pairwise weight $c_{ij} = c_i c_j$ greatly simplified the learning and inference process, but at the cost of model expressibility (correlations between depth values are not captured). However several extensions can be made to our model to add more expressiveness while retaining efficiency in both learning and inference. In this section we propose some potential improvements to our model.

We first consider our locally connected pairwise energy terms:

$$E_{lc}(y) = \sum_{i,k} c_i^d c_k^d \big((y_k - y_i) - (y_k^d - y_i^d)\big)^2 \tag{5.17}$$

where $k \in \{+u(i), +v(i)\}$ denotes pixel locations to the right of and below pixel i in the image plane (forming a grid-connected graph). We can introduce more flexibility for the neighborhood size in order to account for longer range pairwise depth relationships. Under the typical Gaussian process assumption, the error in pixel depth ratios is roughly proportional to the distance between the pixels. Under this assumption long range depth relationships are likely to be less accurate than short range ones. The optimal neighborhood size would however vary depending on the quality of y^d. We can explicitly model this assumption by multiplying $c_i^d c_j^d$ with a Gaussian RBF (radial basis function) kernel f_{ij}^d:

$$f_{ij}^d = \frac{1}{Z_f} e^{-|x_j - x_i|_2^2 / 2\theta_f^2} \tag{5.18}$$

where $|x_j - x_i|_2$ computes norm of pixel location difference, $Z_f = \sum_{j \neq i} e^{-|x_j - x_i|^2 / 2\theta_f^2}$ is a normalization constant, and θ_f is a tunable parameter. This general form gives one the flexibility to tune the pairwise neighborhood size by varying θ_f for best results. Note that setting θ_f to be a very large value makes the local pairwise term similar to the fully-connected one, thus making the extra fully-connected term redundant.

We can also draw inspiration from the models used in the cross (or joint)-bilateral filter (Petschnigg et al. [2004]) and Colorization (Levin et al. [2004]). Both methods are based on a simple but surprisingly effective pairwise affinity model that assumes neighboring depths (the input signal) within a local region should have similar value if their color (the guiding signal) is similar.

We first formally introduce the model for the cross-bilateral filter (Petschnigg et al. [2004]):

$$E_{bilateral}(y) = \sum_{i,j} f_{ij} g_{ij}^{gauss} (y_j - y_i)^2 \tag{5.19}$$

where f_{ij} is the same pixel location difference-based Gaussian RGB kernel that we introduced earlier, and g_{ij}^{gauss} is given by:

$$g_{ij}^{gauss} = \frac{1}{Z_g} e^{-|I_j - I_i|_2^2 / 2\theta_g^2} \tag{5.20}$$

which is an image color difference based Gaussian RBF kernel, where $|I_j - I_i|_2$ computes norm of pairwise image color difference, $Z_g = \sum_{j \neq i} e^{-|I_j - I_i|^2 / 2\theta_g^2}$ is a normalization constant, and θ_g is a tunable parameter. Note that the cross-bilateral filter is a simple variant of the bilateral filter originally proposed in Tomasi and Manduchi [1998] for edge-preserving smoothing of images, with the difference being that in the original bilateral filter, g_{ij}^{gauss} is a function of y instead of I.

Next we introduce the model for the colorization method (Levin et al. [2004]):

$$E_{colorization}(y) = \sum_i \sum_{j \in \mathcal{N}(i)} g_{ij} (y_j - y_i)^2 \tag{5.21}$$

The authors in Levin et al. [2004] try out two pairwise affinity kernels in place of g_{ij}. In their first version, $g_{ij} = g_{ij}^{gauss}$ of the cross-bilateral filter. In their second version, $g_{ij} = g_{ij}^{corr}$ is a correlation kernel derived from the assumption that the input signal, e.g. depth, within a local window $\mathcal{N}(k)$ is a *linear* function of guiding signal, e.g. image color (He et al. [2013]):

$$g_{ij}^{corr} = \frac{1}{|\mathcal{N}(k)|^2} \sum_{k:(i,j) \in \mathcal{N}(k)} 1 + (I_i - \mu_k)^T \frac{1}{\Sigma_k^2} (I_j - \mu_k) \tag{5.22}$$

where μ_k and Σ_k are the mean vector and covariance matrix of a patch in the image, formed by the pixels in a local window $\mathcal{N}(k)$ around k, and $\sum_{j \in \mathcal{N}(i)} g_{ij}^{corr} = 1$. Based on the experiments in Levin et al. [2004] both versions of g_{ij} produced visually similar results with hyper-parameters manually tuned (but they chose the correlation kernel for the results shown in the paper). They use an optimal window size of 3x3 pixels (also extending this window to the temporal domain when provided with a video sequence with optical flow). Their goal was to produce color images from intensity images, therefore I represented intensity (grayscale value), and μ and Σ_i were scalars. In He et al. [2013] the correlation filter was shown to perform better (produce sharper edges) for the task of color image sharpening when the guiding signal I is in color rather than in grayscale.

Note that the guided filter in He et al. [2013] performs an explicit local mean filtering on y, while the colorization method (Levin et al. [2004]) performs mean filtering implicitly via global optimization of the energy with respect to y. Both methods however are based on the same correlation filter-based model. In He et al. [2013] it was shown that local explicit filtering is more prone to halo effects caused by low-contrast edges in the guide image I, while global optimisation-based methods are more prone to intensity shifts in the filtered solution over large regions. He et al. [2013] also point out that the Gaussian filter is susceptible to the gradient reversal problem which introduces unwanted edges but apart from that the results are very similar to that of the correlation filter. The main advantage of the correlation filter is that it can be applied through successive mean filter updates and *exact* solutions can be found in $O(N)$ time, even for a large window size (He et al. [2013]). However efficient approximate inference/filtering methods using the Gausssian filter also exist for large neighborhood sizes and filter dimensions (Adams et al. [2010]; Barron and Poole [2016]; Krähenbühl and Koltun [2011]).

It is interesting to note that the role of both the kernels g_{ij}^{gauss} and g_{ij}^{corr} is to define the mapping from I into y, the difference being in the underlying assumption of the mapping (the latter assumes a piecewise linear mapping). Hence these models would be more effective if we were to *learn* some pixelwise mapping from $I \in \mathcal{R}^3$ into an intermediate K dimensional feature representation $g \in \mathcal{R}^K$ — replacing I in Equations (5.20) and (5.22) — such that the respective underlying assumption and model is satisfied. This also makes the choice of underlying model less important. This is in fact the direction taken by recent CNN-based methods: the Deep Bilateral Filter (Gharbi et al. [2017]) and the Deep Guided Filter (Wu et al. [2018]), which we draw further inspiration from.

It is not completely necessary to add the pairwise affinity model to our model, as the highly non-linear relationship between depth and color may have been already implicitly learned by the CNN depth predictors. Nevertheless, a well-learned pairwise affinity model is likely to contribute to the overall performance rather than hinder it. Also, it is reasonable to assume that it is easier to learn and generalize a mapping from I to g than it is to learn and generalize and mapping from from I to y. Having an explicit pairwise affinity model (which already performs reasonably well even when the input is raw pixel color) allow for some guarantee of generalizablility for input very different to that of the training set, when depth predictions are likely to be less accurate.

Therefore we introduce the following local pairwise affinity-based energy:

$$E_{la}(y) = \sum_{i,j} f_{ij}^a g_{ij}^a c_i^a c_j^a (y_j - y_i)^2 \tag{5.23}$$

where f_{ij}^a is the same pixel distance based Gaussian kernel, g_{ij}^a can be either the correlation filter or the Gaussian filter parameterized by a *learnable* (Gharbi et al. [2017]; Wu et al. [2018]) K dimensional pixelwise feature representation g^a instead of I, and $c_i^a c_j^a$ is an approximation to the confidence of the pairwise term which is analogous to $c_i^d c_j^d$ and can be *learned* using our method proposed in Section 5.4.3. Note that we chose the pairwise pixel-distance-based Gaussian RBF kernel f_{ij}^a as recommended in He et al. [2013] to gradually reduce filter strength with increasing distance to i uniformly in all directions rather than the square window based neighborhood of the colorization method and guided filter which tend to apply filtering less uniformly, leading to stronger filtering in horizontal and vertical directions than others. It is important to reiterate that the need for both kernels f_{ij}^a and g_{ij}^a arise due to our relaxation of the learnable pairwise confidence weight, i.e. $c_{ij}^a = c_i^a c_j^a$.

We can also multiply g_{ij}^d with the pairwise weights of our locally connected term E_{lc} giving rise to the overall pairwise weight: $c_{ij}^d = f_{ij}^d g_{ij}^d c_i^d c_j^d$. Similar to before, g_{ij}^d can be either the correlation filter or the Gaussian filter parameterized by a *learnable* K dimensional pixelwise feature representation g^d instead of I. The role of the kernel g^d_{ij} in this case is to enforce *errors in* y (rather than y itself) to be similar around a local window if their appearance is similar. Note that this is a more realistic assumption. The introduction of g^d_{ij} will compensate for the expressibility lost due to the approximation $c^d_{ij} = c^d_i c^d_j$, and further compensate for possible errors in confidence prediction.

We can now re-write our modified locally connected pairwise term as:

$$E_{lc}(y) = \sum_{i,j} f^d_{ij} g^d_{ij} c^d_i c^d_j \big((y_j - y_i) - (y^d_j - y^d_i)\big)^2 \tag{5.24}$$

Our log depth map solution y will now depend on the result of minimizing the following overall energy:

$$E(y) = \alpha E_u(y, y^s, c^s) + \beta E_{fc}(y, y^d, c^d) + \gamma E_{lc}(y, y^d, c^d, g^d) + \eta E_{la}(y, c^a, g^a) \tag{5.25}$$

where $(\alpha, \beta, \gamma, \eta) > 0$ are the hyper-parameters of our model.

5.4.2 An Inverse Depth Parameterization

In this chapter we parameterized our observation model and depth prior model terms in log depth space, as nodes and edges of a CRF respectively. Firstly, this parameterization introduced homoscedasticity, i.e. more uniform variance throughout the solution space, which favours the least squares estimator. Secondly in this parameterization it was possible to formulate a scale-invariant pairwise depth error measure that is the norm of a simple *linear function with respect to the solution.* Both criteria can be satisfied in inverse depth parameterization as well.

Consider the following equation involving two inverse depth values ρ_i and ρ_j in the solution, and their corresponding CNN predicted values ρ^d_i and ρ^d_j:

$$\frac{\rho_j}{\rho_i} = \frac{\rho^d_j}{\rho^d_i} \tag{5.26}$$

Rearranging the terms:

$$\rho_j - \rho_i \frac{\rho^d_j}{\rho^d_i} = 0 \tag{5.27}$$

Therefore, we can minimise the following energy with respect to ρ as the solution is independent of the scale of the predicted inverse depth map ρ^d:

$$E_{lc}(\rho) = \sum_{i,j} f^d_{ij} g^d_{ij} c^d_i c^d_j \Big(\rho_j - \rho_i \frac{\rho^d_j}{\rho^d_i}\Big)^2 \tag{5.28}$$

Our overall energy can be re-formulated in inverse depth parameterization as:

$$\begin{aligned}
E(\rho) &= \alpha E_u(\rho, \rho^s, c^s) + \beta E_{fc}(\rho, \rho^d, c^d) + \gamma E_{lc}(\rho, \rho^d, c^d, g^d) + \eta E_{la}(\rho, c^a, g^a) \\
E_u(\rho) &= \sum_i^N c_i^s (\rho_i - \rho_i^s)^2 \\
E_{fc}(\rho) &= \frac{1}{2N} \sum_{i,j} c_i^d c_j^d \Big(\rho_j - \rho_i \frac{\rho_j^d}{\rho_i^d}\Big)^2 \\
E_{la}(\rho) &= \sum_{i,j} f_{ij}^a g_{ij}^a c_i^a c_j^a (\rho_j - \rho_i)^2
\end{aligned} \tag{5.29}$$

Note that $E_{fc}(\rho)$, just as $E_{fc}(y)$, can be written in a linear-time computation form as follows:

$$E_{fc}(\rho) = \frac{1}{2N} \sum_i c_i^d \sum_j c_j^d \rho_j^2 - \frac{1}{N} \sum_i c_i^d \Big(\frac{\rho_i}{\rho_i^d}\Big) \sum_j c_j^d \rho_j \rho_j^d + \frac{1}{2N} \sum_i c_i^d \Big(\frac{\rho_i}{\rho_i^d}\Big)^2 \sum_j c_j^d (\rho_j^d)^2 \tag{5.30}$$

5.4.3 Learning Scale-Invariant Depth Prior Confidences

Recall that previously in this chapter we proposed to learn the confidence of a sparse depth map/CNN depth map in a heuristic manner via minimization of Euclidean distance to an explicitly computed relative confidence map c^s based on scale-invariant depth error:

$$L^s = \frac{1}{|b|} \sum_i^N b_i^s (\hat{c}_i^s - c_i^s)^2 \tag{5.31}$$

where

$$c_i^s = b_i^s e^{-\lambda^s |E_i^s|} \tag{5.32}$$

and E_i^s is the scale-invariant error for a depth value in pixel i which we defined as:

$$\begin{aligned}
E_i^s &= (\alpha^s + \beta^s) b_i^s (y_i^s - y_i) \\
&- \frac{\beta^s}{|b|} \sum_j^N b_j^s (y_j^s - y_j) \\
&+ \gamma^s \sum_{j \in \mathcal{N}(i)} b_i^s b_j^s \big((y_j^s - y_i^s) - (y_j - y_i)\big)
\end{aligned} \tag{5.33}$$

In this section we propose an alternate way to learn the relative confidences $c_i, \forall i$ of a noisy depth map (or any input signal) in a more *implicit* manner, by maximizing the likelihood of the model with respect to c_i in the pairwise domain (c_i here means c_i^s, c_i^d, c_i^a, etc). More specifically, we intend to find the pairwise confidences $c_{ij} = c_i c_j, \forall (i, j)$ that maximize the likelihood of all the edges of a CRF, which we model as a multi-variate Gaussian PDF (probability density function) parameterized by some *pairwise error measure* (based on the input variables) with a *mean of* 0 (or close to 0). Note that if we were to remove our pairwise confidence relaxation $c_{ij} = c_i c_j, \forall (i, j)$ and directly estimate $c_{ij}, \forall (i, j)$ instead of just $c_i, \forall i$ this would require $O(N^2)$ many variables to estimate and store.

Analytically we show that solving for $c_i, \forall i$ in this problem leads to solving an overdetermined system of linear equations (more constraints than unknowns), but the least squares solution is a confidence map $(c_i, \forall i)$, where each c_i provides a measure of the *relative correctness* of the corresponding value in the depth map (or input signal) — i.e. a relative confidence map.

While our previous heuristic approach (Equation (5.32)) also computes a relative confidence map based on scale-invariant error (and serves as a good approximation), we show that the more correct least squares solution for c_i that best satisfies our model is not computed in the same way. In fact, the more ideal solution for c_i requires $O(N^2)$ operations to compute. Therefore an implicit learning scheme is the more efficient approach if we want to learn to regress for this solution.

This is because (as we will show later on) the required gradient computation for implicit learning can done efficiently in $O(N)$ time. If the pairwise confidence weight is multiplied with a d-dimensional Gaussian RBF kernel (Section 5.4.1) the gradient can still be approximately computed in $O(Nd)$ time. While it remains to be shown empirically, it is very likely that through our approach a CNN is capable of implicitly learning to regress a solution that is close to the least squares estimate for $c_i \forall i$, instead of having to minimise the Euclidean distance to a pre-computed confidence map based on the least squares solution.

As in our previous objective function (Objective (5.31)), our new loss provides the benefit of not having to pre-process the input or scale it to match groundtruth scale. Furthermore, our new approach is likely to be more robust to noise in the observations as confidences are implicitly learned to maximize the likelihood of the model, automatically giving lower weight to variables with large error. Our previous heuristic to approximate scale-invariant error on the other hand is very sensitive to outliers (Equation (5.33)). Our approach is useful in the case of learning confidences for monocular SLAM depth maps, which have unknown scale and consists of outliers. It is also somewhat useful in the case of learning confidences of CNN depth maps where the overall scene scale is wrongly predicted, especially for images very different to the training set. However if the noisy depth map is in a known scale then our approach is not very attractive (although still applicable), and it may be better to maximize the likelihood of the nodes of a CRF rather than the edges, similar to existing methods (Kendall and Gal [2017]).

More broadly, our new approach is useful in problems where: (1) the scale factor aligning a groundtruth and observed continuous signal (such that the mean error is close to 0) is a function of the point-wise confidences, and these confidences are not known and we want to implicitly learn these confidences from the data without explicitly solving for them (as it is computationally expensive) or having to pre-process the data, and (2) knowing relative confidence (variance) of the variables in the data is more important (or sufficient) than knowing their absolute variance, and/or (3) we simply want an efficient means of implicitly learning approximate confidences of the *pairwise edges* of any continuous CRF model — with only $O(N)$ learnable parameters, N being the number of unary variables — given that the distributions of the *edges* are (approximately) multi-variate Gaussian with mean ≈ 0, are independent, and the number of edges is very large.

We hereby detail our approach and leave experimentation for future work. We start by assuming that we are given with (or able to formulate from the input variables) pairwise errors $r_{ij}, \forall(i,j)$, that are Gaussian distributed and independent with mean of 0 and standard deviations $\sigma = \{\sigma_{ij}, \forall(i,j)\}$. Some examples of r_{ij} from our work are the pairwise scale-invariant log depth error: $r_{ij} = (y_j - y_i) - (y_j^d - y_i^d)$, the pairwise scale-invariant inverse depth error: $r_{ij} = \rho_j - \rho_i \rho^d_j/\rho^d_i$, the pairwise log depth smoothness error: $r_{ij} = (y_j - y_i)$, and the pairwise inverse depth smoothness error: $r_{ij} = (\rho_j - \rho_i)$. Note that the latter two pairwise errors are used in our newly proposed local pairwise affinity energy E_{la} (Sections 5.4.1 and 5.4.2) and are best modeled with a Laplace distribution. However the kernel weight g_{ij} reduces the heteroscedasticity in their original distribution and makes their resulting distribution close to a Gaussian. In order to learn the parameters σ of the Gaussian CRF model in the pairwise domain we can minimize the following negative log likelihood objective:

$$\min_{\sigma} \sum_{i,j \neq i} -ln \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp(-\frac{1}{2\sigma^2_{ij}} r^2_{ij}) \tag{5.34}$$

In its original form the expression in Objective (5.34) has exponentially many variables to learn for σ. Therefore we can relax the dimensionality of the learning problem by letting $\sigma^2_{ij} = 1/(c_i c_j)$, where $c = \{c_i, \forall i\}$ represents the confidences of the unary variables forming the pairwise errors. We also incorporate the improvements to our model as we discussed in Section 5.4.1, and therefore we re-write $\sigma^2_{ij} = 1/(f_{ij} g_{ij} c_i c_j)$. The objective can now be written as:

$$\begin{gathered} \min_{c} \sum_{i,j \neq i} -ln \sqrt{\frac{f_{ij} g_{ij} c_i c_j}{2\pi}} \exp(-\frac{f_{ij} g_{ij} c_i c_j}{2} r^2_{ij}) \\ \min_{c} \sum_{i,j \neq i} \frac{f_{ij} g_{ij} c_i c_j}{2} r^2_{ij} - \frac{1}{2} \sum_{i,j \neq i} ln \frac{f_{ij} g_{ij} c_i c_j}{2\pi} \end{gathered} \tag{5.35}$$

Note that the kernel weights and neighbourhood region in the general case are not restricted to that in the above objective. For example, we can choose to set $f_{ij} = 1, g_{ij} = 1, \forall (i,j)$, and change the neighborhood region $j \in \mathcal{N}(i)$ arbitrarily, based on a valid assumption about the nature of the pairwise errors r_{ij}.

Simplifying the above expression and dividing it by N-1 (in order to normalize for the average error per pixel based on its N-1 neighbours) yields:

$$\min_{c} \frac{1}{N-1} \sum_{i,j \neq i} \frac{f_{ij} g_{ij} c_i c_j}{2} r^2_{ij} - \sum_{i} ln c_i + K \tag{5.36}$$

where $K = -\frac{1}{2(N-1)} \sum_{i,j \neq i} ln \frac{f_{ij} g_{ij}}{2\pi}$.

Differentiating the expression in Objective (5.36) with respect to c_i gives the following gradient expression:

$$\frac{1}{N-1} \sum_{j \neq i} f_{ij} g_{ij} c_j r^2_{ij} - \frac{1}{c_i} \tag{5.37}$$

Although in practice we don't need to explicitly solve for c_i, as our training is based on stochastic gradient descent-based optimization for solving Objective (5.36) (summed over all training instances), it is still of interest to understand what the quantity c_i actually represents. We can first solve for the variables σ_{ij} which gives N(N-1)/2 unique constraints for the N variables in c. Solving this overdetermined system of equations for c using least-squares gives a closed form solution for c_i:

$$c_i^* = \frac{\prod_{k\neq i, j\neq i} f_{jk} g_{jk}(r_{jk})^{2/(N-1)(N-2)}}{\prod_{j\neq i} f_{ij} g_{ij}(r_{ij})^{2/(N-1)}} \tag{5.38}$$

From the above expression it can be seen that the least squares solution for c_i is a measure of relative correctness of the i^{th} variable in the signal. It is based on the ratio of the product of pairwise residuals not including variable i to the product of pairwise residuals that include it. If variable i is an outlier then all the pairwise residuals that are based on it are likely to be large thus making the denominator large, leading to a lower confidence, and vice versa. Explicitly computing this expression requires $O(N^2)$ operations and can be avoided through implicit learning by solving Objective (5.36) (summed over all training instances) via stochastic gradient descent. With a d-dimensional Gaussian RBF kernel, the gradients can be approximately computed in $O(Nd)$ time by using the fast Permutohedral lattice filtering scheme of Adams et al. [2010] coupled with some other approximations (Krähenbühl and Koltun [2011]).

Dealing with Sparse Signals: Most often the noisy data and/or the groundtruth signal is sparse, e.g. the sparse SLAM depth map. The pairwise terms formed by invalid unary variables will drive the pairwise variance to large values making learning unstable. Hence such pairwise terms must not be modeled as part of the multi-variate Gaussian distribution. An easy fix for this is to introduce a binary mask b, where b_i is 0 if either the data point i or groundtruth point i is missing, into Objective (5.36), and re-write it as:

$$\min_c \frac{1}{|b|-1} \sum_{i,j\neq i} \frac{f_{ij} g_{ij} b_i b_j c_i c_j}{2} r_{ij}^2 - \sum_i b_i ln c_i + K \tag{5.39}$$

where $K = -\frac{1}{(|b|-1)} \sum_{i,j\neq i} b_i b_j ln \frac{f_{ij} g_{ij}}{2\pi}$. However if we are just aware of the sparsity b_{ij} of edge variables in the CRF, i.e. r_{ij} then we can exclude only these pairwise edges by multiplying each of the terms in Objective (5.35) with b_{ij}.

Normalizing c: The values of the learned $c_i, \forall i$ may sometimes be very large depending on the nature of the residual errors r_{ij}. After the CNN confidence predictor is learned we can simply normalize c by dividing it with a global constant such that it is in a manageable range. For instance the normalization could be done by: $c_i \leftarrow c_i/\sigma_c$, where σ_c is the standard deviation of all the predicted c_is in the training set. This would make it easier to tune the global hyper-parameters of our just-in-time reconstruction CRF (α, β, γ, etc.) during inference.

Modifying the Unary Weight of Our Just-In-Time Reconstruction CRF: Based on the understanding we gained about c_i in this section, it is apparent that c_i represents a measure that is proportional to standard deviation rather than variance. Recall that we previously defined the unary term of our just-in-time reconstruction CRF to be:

$$E_u = \sum_i c_i^s (y_i - y_i^s)^2 \tag{5.40}$$

In the above formulation we assumed c_i^s to be proportional to variance. The following simple modification of our unary term would make the learnt c_i a better fit for our unary model:

$$E_u = \sum_i (c_i^s)^2 (y_i - y_i^s)^2 \tag{5.41}$$

Special Cases: Some special cases allow Objective (5.36) and its gradients to be computed *exactly* in $O(N)$ time. For example consider the case where $f_{ij} = 1, g_{ij} = 1, \forall(i,j)$ and $r_{ij} = (y_j - y_i) - (y_j^d - y_i^d)$. Objective (5.36) can then be equivalently expressed as:

$$\min_c \frac{1}{N-1} (\sum_i c_i r_i^2 \sum_j c_j - (\sum_i c_i r_i)^2) - \sum_i ln c_i + K \tag{5.42}$$

where $r_i = y_i - y_i^d$ in this case. An exact $O(N)$ time expression can also be derived with our proposed inverse-depth parameterized pairwise error: $r_{ij} = \rho_j - \rho_i \rho_j^d / \rho_i^d$ as discussed in Section 5.4.2.

Relation to The Scale-Invariant Loss in Eigen et al. [2014]: Referring to the expression in Objective (5.42), notice how setting $c_i = 1, \forall i$ (i.e. assuming unit variance for all pairwise errors, making lnc_i disappear), the objective becomes identical to the scale-invariant loss objective proposed in Eigen et al. [2014], the latter being minimised w.r.t. y^d instead of c, along with a unary term to define scale. Therefore our formulation can be seen as a much more flexible generalization of the loss in Eigen et al. [2014].

We can show why a unary term is needed if we were to learn y^d by differentiating the expression in Objective (5.42) with respect to y^d:

$$\frac{2c_i}{N-1} \sum_{j \neq i} c_j r_{ij} \tag{5.43}$$

We can now equate the above expression to 0, obtaining N-1 unique equations for the N variables in y^d:

$$y_i^{d*} = y_i + \sum_{j \neq i} c_j (y_j^{d*} - y_j) / \sum_{j \neq i} c_j \tag{5.44}$$

Notice that there are infinite solutions for predicted log depth y^d all of which are some variable additive offset apart from the groundtruth log depths y (a variable multiplicative scale factor apart in depth space). This offset is variable because it is a weighted average of log depth errors of all the *other* variables in the log depth map, which can change depending on the state of the other variables. Note however that, when N is large, this variable offset is very similar (almost the same) for all the variables in the log depth map, and we can regard it as a *mutual offset*. This means that we are learning an *up-to-scale* depth map with no prior on the scale. Hence why, in order to regularize the log depth predictions to be close as possible to the groundtruth in absolute value, we need to add a unary term $\frac{\epsilon}{2} \sum_i^N c_i^2 (y_i - y_i^d)^2 - \epsilon \sum_i^N ln \frac{c_i}{2\pi}$ to Objective (5.42), where ϵ is a tunable parameter which controls the scale-invariance of the solution.

5.4.4 A Unified Model for Just-In-Time Reconstruction

We now consider the case we are given a set of M nearby images with some baseline, one of which is the keyframe we want to reconstruct, as well as a sparse inverse depth map ρ^s, e.g. from SLAM/a sparse depth sensor. As before, the desired outcome is an accurate dense depth map for the keyframe, that is generated in the most efficient manner. Our aim here is to make efficient use of all of the available observations as well as information from the stored compact map. For this purpose we also re-visit our learned multi-view matching cost observation model proposed in Chapter 4, and our learned surface normal prior model proposed in Chapter 3.

Combining our learned models proposed so far, and using inverse depth as the parameterization of choice, leads to the following unified model for just-in-time reconstruction:

$$\begin{aligned} E(\rho) = &\frac{1}{\lambda} E_\phi(\rho, f_{ref}, f_i, ..., f_M) + \\ &E_{\hat{n}}(\rho, \hat{n}, c^{\hat{n}}) + \\ &\alpha E_u(\rho, \rho^s, c^s) + \\ &\beta E_{fc}(\rho, \rho^d, c^d) + \\ &\gamma E_{lc}(\rho, \rho^d, c^d, g^d) + \\ &\eta E_{la}(\rho, c^a, g^a) \end{aligned} \tag{5.45}$$

where,

$E_\phi(\rho) = \sum_{i=1}^{N} \frac{(c_i^\phi)^2}{M} \sum_{m=1}^{M} \lvert f_{ref}(u_i) - f_m(\pi(T_{mr}\pi^{-1}(u_i,\rho_i)))\rvert_1$	:	*Learned* Multi-View Matching Cost Observation Model [Full Form] $f_{ref} \leftarrow I_{ref}$ $f_m \leftarrow I_m, m \in \{1, ..., M\}$ $c^\phi \leftarrow I_{ref}$
$E_\phi(\rho) \approx \sum_{i=1}^{N} (c_i^\phi)^2 (\rho_i - \rho_i^{mv})^2$	:	[Approximate Convex Form]
(Chapter 4, Sections 4.5.1 and 4.5.2.)		
$E_{\hat n}(\rho) = \sum_{i,j \in \mathcal{N}(i)} b_i^{\hat n}(c_i^{\hat n})^2 (\langle \hat n_i, \tilde x_i\rangle \rho_j - \langle \hat n_i, \tilde x_j\rangle \rho_i)^2$	:	*Learned* Surface Normal Prior Model [Quadratic Form] $\hat n \leftarrow I_{ref}$ $c^{\hat n} \leftarrow I_{ref}$ $b^{\hat n} \leftarrow I_{ref}$
$E_{\hat n}(\rho) = \sum_{i=1}^{N} b_i^{\hat n}(c_i^{\hat n})^2 \left\lvert \begin{matrix} \langle \hat n_i, \tilde x_i\rangle \rho_p - \langle \hat n_i, \tilde x_p\rangle \rho_i \\ \langle \hat n_i, \tilde x_i\rangle \rho_q - \langle \hat n_i, \tilde x_q\rangle \rho_i \end{matrix} \right\rvert_\epsilon$	:	[Huber Norm Form]
(Chapter 3, Sections 3.3.2 and 3.5.2.)		
$E_u(\rho) = \sum_{i=1}^{N} (c_i^s)^2 (\rho_i - \rho_i^s)^2$	:	*Learned* Sparse Map Observation Model $c^s \leftarrow I_{ref}$
(Chapter 5, Sections 5.2.1, 5.4.2 and 5.4.3.)		
$E_{fc}(\rho) = \frac{1}{2N} \sum_{i,j} c_i^d c_j^d \Big(\rho_j - \rho_i \frac{\rho_j^d}{\rho_i^d}\Big)^2$	:	*Learned* Fully-Connected Depth Prior Model $\rho^d \leftarrow I_{ref}$ $c^d \leftarrow I_{ref}$
(Chapter 5, Sections 5.2.1 and 5.4.2.)		
$E_{lc}(\rho) = \sum_{i,j} f_{ij}^d g_{ij}^d c_i^d c_j^d \Big(\rho_j - \rho_i \frac{\rho_j^d}{\rho_i^d}\Big)^2$	:	*Learned* Locally-Connected Depth Prior Model $\rho^d \leftarrow I_{ref}$ $c^d \leftarrow I_{ref}$ $g^d \leftarrow I_{ref}$
(Chapter 5, Sections 5.2.1, 5.4.1 and 5.4.2.)		
$E_{la}(\rho) = \sum_{i,j} f_{ij}^a g_{ij}^a c_i^a c_j^a (\rho_j - \rho_i)^2$	:	*Learned* Locally-Connected Affinity Prior Model $c^a \leftarrow I_{ref}$ $g^a \leftarrow I_{ref}$
(Chapter 5, Sections 5.4.1 and 5.4.2.)		

Note that the above formulation incorporates all improvements and fixes to our "just-in-time reconstruction" model proposed in this chapter as well as those proposed in Chapters 3 and 4. Apart from E_ϕ (full form), all the energy terms are *convex functions* with respect to ρ, the *inverse depth map solution* of the keyframe. We have presented our normal prior model in quadratic form as well as Huber norm form. Recall that we used the Huber norm version for our frameworks in Chapter 3 and 4. However, the learned normal confidence map $c^{\hat{n}}$ and the learned depth discontinuity mask $b^{\hat{n}}$ that we introduced in Chapter 3, Section 3.5.2 allow the error distribution of the normal prior model to be modeled as a multi-variate Gaussian. This makes the quadratic form a reasonable choice. The Huber norm version, which assumes a Gaussian distribution for smaller errors and a Laplace distribution for errors beyond a certain threshold, is still a good alternative, especially if the learned $c^{\hat{n}}$ fails to provide a low enough confidence at depth discontinuities and/or if there is no learned depth discontinuity mask $b^{\hat{n}}$ available.

All the other models that are written in quadratic form can be replaced with a robust norm like Huber as well. However application of a robust norm on the fully connected pairwise terms can be inefficient. For example the Iteratively Re-Weighted Least Squares estimator requires computation of a normalization weight for each edge term, and for the primal dual method the number of dual variables to store and optimize grow linearly with the number of pairwise edges. Nevertheless the need for robust norms is reduced as our learnable unary and pairwise confidence weights make all of our model error distributions close as possible to a multi-variate Gaussian.

For the convex optimization problem with all terms in the quadratic form, standard methods can be used like the iterative conjugate gradient method as discussed in this chapter. However, if the Huber norm form is used, methods like iterative conjugate gradient cannot be used. Also, the second derivative of the Huber norm is not continuous which prevents certain optimization methods (e.g. Newton's method) from being used. The efficient primal dual method is preferred in such cases.

If the full form of E_ϕ is used, similar to the algorithm used in DTAM (Newcombe et al. [2011]) and Chapters 3 and 4, we can introduce a coupling term and alternate between solving a convex and non-convex energy, gradually forcing the auxiliary variables of the non-convex energy to be close to the solution of the convex energy. The minima of the non-convex energy can be found through a bounded brute-force search (Newcombe et al. [2011]). However, as discussed at the end of Chapter 4, since our learned-features-based matching cost energy has fewer local minima and increased convexity (in comparison to the photometric error-based matching cost energy), it is worth exploring a gradient-based optimization scheme involving linearization of the matching cost at the current solution. Furthermore if the convex approximation to E_ϕ is used instead, the whole optimization problem becomes convex and it can potentially offer a significant speedup in keyframe reconstruction with only a slight decrease in accuracy.

This unified model for just-in-time reconstruction serves as a taxonomy and summary of the learnable probabilistic models proposed in this thesis for geometric reconstruction. It is important to note that all the terms of the unified model have a confidence associated to them, which not only allows for more accurate information fusion, but also allows us to obtain a per-pixel confidence for the estimated keyframe inverse depth map ρ that satisfies the overall model. This confidence map will come in useful, for example, during volumetric fusion of depth maps.

5.5 Conclusion

In this chapter we advocated “just-in-time reconstruction”, a flexible and efficient approach to dense reconstruction that utilizes a sparse map and a higher-level form of scene understanding based on a single live image, to generate a dense reconstruction *on-demand*, that is consistent with the sparse map. This approach is especially useful during large-scale mapping where it’s inefficient to store detailed information about the map and all the corresponding images. We modeled the task as inference over a novel fully connected CRF model, with nodes anchoring the solution to the sparse map and the scale-invariant edges enforcing pairwise depth relationships based on depth predictions of a deep neural network, that are predicted from a live RGB image. The CRF model was also parameterized by point-wise confidences of both the sparse map and dense depth predictions, and these confidences were predicted using CNNs, given the depth maps and the live RGB image as input. This form of probabilistic data fusion allowed the solution to be less sensitive to the presence of erroneous depths, and a simple relaxation of the pairwise confidence weight enabled efficient inference of the solution. We applied our method to perform real-time image-guided inpainting of sparse maps obtained from three different sources: a sparse monocular SLAM framework, Kinect and LIDAR. Our method is very general and applicable for fusing depth information from multiple modalities. We also proposed a unified formulation that additionally incorporates the surface normal prior of Chapter 3, as well as other information such as multi-view consistency based a set of nearby images, and pairwise affinity-based implicit depth filtering. Therefore this work opens up more useful applications to be explored.

Bibliography

A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010.

J. T. Barron and B. Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016.

T. Dharmasiri, A. Spek, and T. Drummond. Joint prediction of depths, normals and surface curvature from rgb images using cnns. *arXiv preprint arXiv:1706.07593*, 2017.

D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

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*, pages 2366–2374, 2014.

J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118, 2017.

M. Hallin. Gauss–markov theorem in statistics. *Wiley StatsRef: Statistics Reference Online*, 2014.

K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1397–1409, 2013.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590, 2017.

A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 66–75, 2017.

P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

L. Ladický, J. Shi, and M. Pollefeys. Pulling Things out of Perspective. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 89–96, jun 2014.

I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23:689–694, 2004.

F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38 (10):2024–2039, 2016.

G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image Inpainting for Irregular Holes Using Partial Convolutions. *arXiv preprint arXiv:1804.07723*, 2018.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

P. L. Odell. Gauss–markov theorem: Overview. *Wiley StatsRef: Statistics Reference Online*, 2014.

G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. In *ACM transactions on graphics (TOG)*, volume 23, pages 664–672. ACM, 2004.

N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. *CoRR*, abs/1708.06500, 2017. URL http://arxiv.org/abs/1708.06500.

B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017.

H. Wu, S. Zheng, J. Zhang, and K. Huang. Fast end-to-end trainable guided filter. In *CVPR*, 2018.

Chapter 6

Conclusion and Future Directions

This chapter provides a summary of the work presented, and a brief discussion of some future directions following up on those discussed in the previous chapters.

6.1 Conclusion

In this thesis we proposed several efficient methods to exploit the learning capabilities of Convolutional Neural Networks (CNNs) to benefit traditional dense geometric reconstruction methods. Particularly, we took advantage of the rich geometric information extracted from a single image via CNNs in the form of surface normals and depths, and formulated prior models parameterized primarily by this learned information — serving as “deeply learned priors” for geometric keyframe reconstruction. We also proposed a method to improve upon the multi-view photoconsistency-based observation model by learning to extract *large receptive field* features from an image that are “good for matching”, so that pixels between two images can be more accurately matched in textureless regions, in the presence of severe appearance distortion. All of this allowed for accurate dense keyframe reconstructions with minimal handcrafting, given noisy, ambiguous and partial observations.

We advocated the concept of “just-in-time” reconstruction where *pairwise relationships in learned depths* from a single image are used to inpaint the corresponding portion of a sparse map, obtained via SLAM or sensory data — facilitating compact storage of a large-scale map (which is useful in practical applications) while improving accuracy and efficiency of online dense reconstruction upon revisiting that map (given only a single image of the map). We further proposed a unified model for “just-in-time” keyframe reconstruction that *jointly accounts for* multi-view geometric matching consistencies based on learned features (given arbitrary number of views overlapping the keyframe) and learned single-view depths, normals and pixel-wise affinities, in addition to depth information from the stored compact map.

Our experiments also showed the importance of incorporating uncertainties of the input information on a per-pixel basis into the reconstruction model and we proposed ways of modeling and learning to predict these uncertainties in a data-dependent manner for each of the prior and observation models that were proposed. Moreover, the experiments throughout this thesis have led us to conclude that *joint reasoning about* scene depths via a good underlying observation model such as the traditional multi-view geometry model — *parameterized by our learned features for matching* — and carefully designed deeply learned prior models — *that are parameterized by neural network predictions about scene geometry where explicit modeling is difficult* — allow for generalizability and accuracy in visual dense reconstructions beyond what could be achieved with either traditional pure handcrafted methods or pure learning based methods alone.

6.2 Future Directions

Monocular Semantic Mapping: In this thesis, we have looked at how scene understanding in the form of learned normals and depths can benefit geometric reconstruction. However *is it possible to now use the recovered geometry to improve semantic scene understanding (i.e. semantic segmentation)?*

Figure 6.1 shows the result of naively fusing (voxel color averaging) RGB color mapped semantic labels (38 classes) predicted using RefineNet (Lin et al. [2017]) trained on the SUN RGB-D dataset (Song et al. [2015]), based on the keyframe depth maps inferred from our framework in Chapter 3. The assignment of colors corresponding to the labels is based on the Pascal VOC toolbox (Everingham et al. [2010]). [1]

This ad-hoc color mapping reduced the dimensionality of the semantic labels from 38 to 3, reducing the amount of storage and computation required for fusing each voxel (each voxel needs to store 3 values instead of the probabilities of 38 classes). However fusing directly in color space is not equivalent to fusing in semantic label space as points nearby in color space don't correspond to points nearby in semantic label space (classes nearby in semantic label space would be very similar to each other and/or are easily confused among one another). Nevertheless, the purpose of our experiment is mainly to see if at all we are able to produce coherent semantic maps at least visually after fusing noisy per-frame semantic label maps, *based on the recovered geometry using our framework*, which indeed is the case.

A proper semantic fusion experiment will use more memory. However an efficient implementation is desirable in practice and such an implementation would involve the use of a more efficient map representation like surfels (or making more efficient use of the TSDF representation by only storing semantic labels for voxels corresponding to the surface), dynamically storing and fusing only the most probable labels for each surface voxel/element, exploiting low-dimensional semantic embeddings, etc.

Future work can also integrate ideas from methods like SemanticFusion (McCormac et al. [2017]) that formulate the labeling problem in 3D as inference over a CRF with pairwise potentials, where class labels of neighboring surfels are taken into account for labeling a particular surface in addition to multi-view surfel label fusion. In Pham et al. [2015], 3D semantic labeling using a CRF with higher order cliques was presented.

While map-based semantic fusion is one way to make use of recovered geometry for improving semantic mapping, future work can further explore the deeper interplay between geometric reconstruction and semantic scene understanding. For instance we can build upon work such as Kundu et al. [2014] that jointly infer the depth and semantic labels of a scene using an explicitly modeled CRF. Semantic segmentation can also potentially be improved with RGB-D input (Eigen and Fergus [2015]) with the depth channel based on the recovered geometry, and/or by using our learned features trained for geometric matching as input to the semantic network as discussed in Chapter 4, Section 4.5.5.

[1]Live video demonstrations of monocular semantic fusion corresponding to Figure 6.1 can be found in: https://youtu.be/OhVD5EP7S_U and https://youtu.be/x8NaoVfS8fk

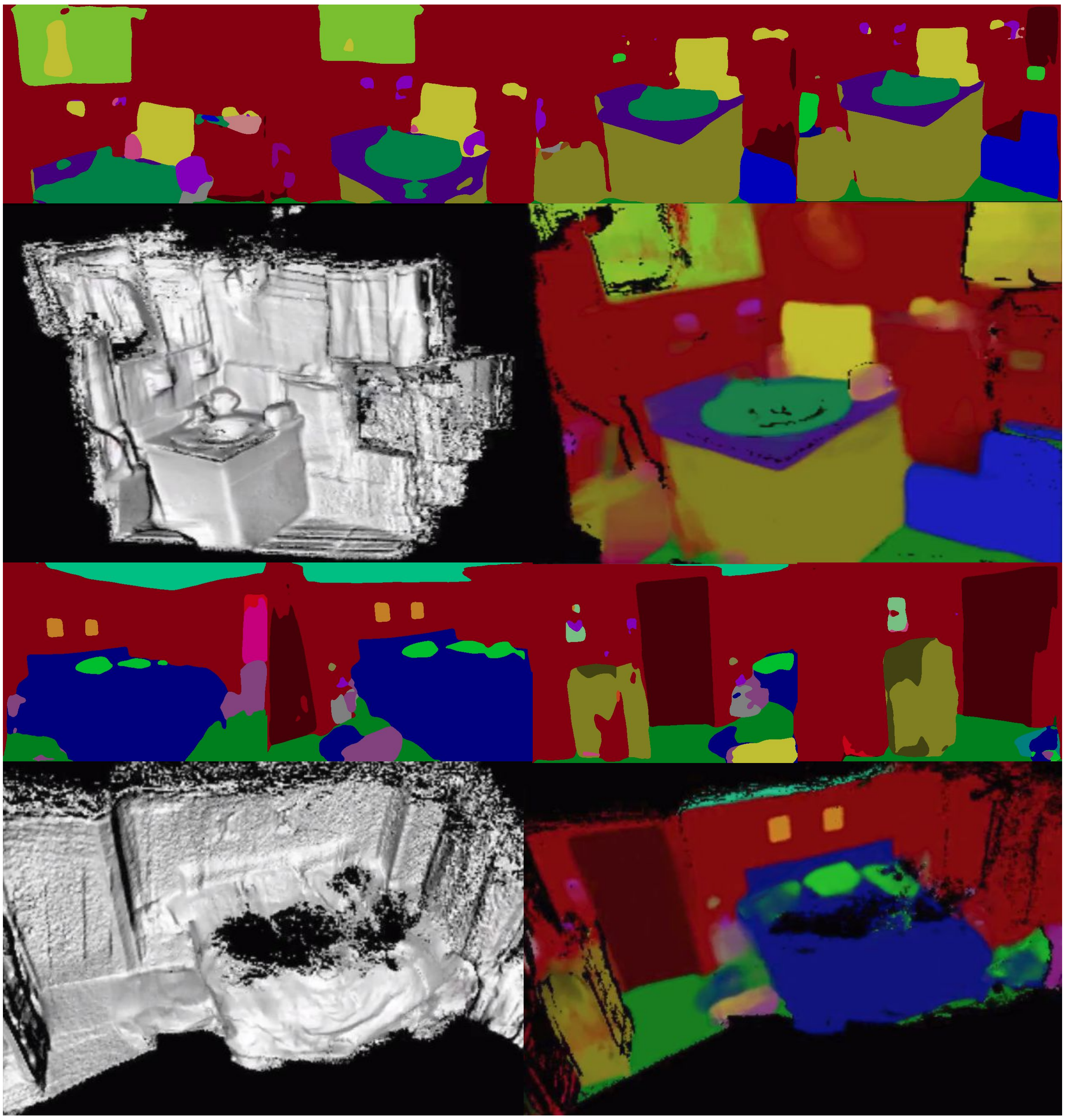


Figure 6.1: This figure shows the result of naively fusing (voxel color averaging) semantic label maps predicted from each keyframe using RefineNet (Lin et al. [2017]) into a TSDF-based voxel grid (Prisacariu et al. [2014]) constructed using the keyframe depth maps estimated by our framework and poses given by monocular tracking. The semantic label maps are RGB color mapped using the Pascal VOC toolbox (Everingham et al. [2010]) where the colors correspond to the 38 classes in the SUN RGB-D dataset. The RefineNet model is trained on the SUN RGB-D dataset. The two sequences depicted are taken from the NYUD-V2 raw test set (Silberman et al. [2012]). The top row in each half of the figure shows a selection of noisy semantic label maps in the respective sequence. As seen in the bottom-right in each half of the figure, even via simple voxel color averaging most of the noise/inaccuracies in the individual semantic label map predictions have been removed. Note that this entire semantic fusion pipeline is based on images from a moving monocular camera.

Deeper Integration of Learning in SLAM: In Chapter 4, Section 4.5.4, we alluded to how our learned features for matching would be beneficial for tasks such as direct image alignment based camera tracking in addition to keyframe reconstruction in the SLAM front-end. While a straightforward approach is to replace handcrafted features for matching with our learned ones in all stages of the monocular visual SLAM pipeline where applicable, there is still more to explore in terms of how *learned priors* can benefit individual stages in the monocular SLAM pipeline like initial two-frame pose recovery, pose graph optimisation, local and global bundle adjustment, place recognition, relocalisation and loop closing.

For instance, during two-frame pose recovery for bootstrapping monocular SLAM, a learned depth map can be used to initialize the map, and the pose between the two frames can be optimized via dense alignment of the two frames utilizing our learned features for matching and learned confidences (Chapter 4, Section 4.5.4), followed by map refinement based on our unified model for "just-in-time" reconstruction (Chapter 5, Section 5.4.4). We can continue to alternate between pose and map refinement until convergence. Initializing the map with a learned depth map in metric scale can also help resolve the scale ambiguity in monocular SLAM.

A practical and efficient SLAM implementation could use a dense direct alignment-based front-end utilizing our learned features and priors (for both mapping and tracking), running alongside a learned feature-based sparse/semi-dense geometric bundle adjustment back-end (for memory and computational efficiency). The front-end could help with data association and map and pose initialization for use in the back-end, and map points could be initialized for regions in the image where the learned feature matching confidence is above a certain threshold. Other components such as pose graph optimisation, place recognition, re-localisation, and loop closing can be added on top, making use of learned features and priors whenever possible thoughout the SLAM pipeline. GPU parallelism can be exploited to add to the efficiency.

Learning Compact Map Representations: Learning can also facilitate the compact storage of a large scale map while preserving rich geometric and semantic information which is useful in practical dense SLAM applications and this complements our "just-in-time" reconstruction approach. Apart from compact map representations such as a sparse set of 3D points, planes, or other geometric primitives such as surfels, we might be able *learn* a suitable low-dimensional manifold that only stores the most relevant information, and decode this compressed information for "just-in-time" reconstruction.

A step in this direction is the work in Cadena et al. where a multi-model autoencoder is used to create a shared code which represented an image, its depth map and semantic label map. The more recent CodeSLAM method (Bloesch et al. [2018]) uses an auto-encoder to compress a dense depth map for a keyframe and store it is as a code. The full resolution dense depth map for a keyframe is recovered by decoding the optimized code conditioned on the keyframe image features. The approach in Bloesch et al. [2018] is limited to encoding keyframe depth maps, and not the full global map, which is a minor downside. The experiments in Cadena et al. were also limited to per-keyframe encoding and decoding. Hence there will be a redundancy in the codes generated for different keyframes. This redundancy can be minimized by reducing the amount of overlap between keyframes. Future work can look at ways of further eliminating this redundancy, and explore other efficient learned compact map representations.

Furthermore, in CodeSLAM, the code is directly used as the parameterization for bundle adjustment given a subset of frames. This is an interesting direction for large-scale dense SLAM as the number of variables for bundle adjustment is greatly reduced while still implicitly optimizing for a dense map. The code is optimized based purely on photometric error, i.e. without any explicit prior in the loss function, as the learned decoder, which is a function of the code and the RGB features, acts as an image-guided depth filter and upsampler, limiting the plausible space of depth map reconstructions. Similar to CodeSLAM, in Tang and Tan [2018] a code is passed through an image-guided learned decoder, and learned feature error is used (instead of photometric error) as the loss function inside the Levenberg–Marquardt algorithm for code optimisation.

The fixed learned decoder in the above methods lessens the importance of having an explicit prior in the loss function for keyframe reconstruction. Nevertheless, it will still be fruitful, for instance, to experiment with our learned feature error and learned priors-based loss function for code optimisation, replacing the photometric error-based loss function in CodeSLAM. Additionally, or alternatively, the keyframe depth map can be optimized in *inverse depth space* by treating the decoded depth map and corresponding predicted confidence map as the “sparse depth map” (ρ^s) and confidence map (c^s) respectively in our unified model for “just-in-time” reconstruction (Chapter 5, Section 5.4.4). The inverse depth solution can be initialized with the decoded depth map for faster convergence. Once optimized, the inverse depth map solution for the keyframe can be encoded and stored compactly until needed again. These are some promising directions that complement our proposed “just-in-time” reconstruction strategy, as the decoded depth map has much richer detail about the scene than a sparse point cloud, enabling rapid dense reconstruction whilst consuming little storage space.

Unconstrained Monocular Non-Rigid Reconstruction: Our unified model for “just-in-time” reconstruction demonstrated how we can combine learned information such as surface normals, depths, good features to match, pixel-wise affinities, and model confidences in a joint-energy minimisation framework (Chapter 5, Section 5.4.4). This unified model however assumed rigidity. Nevertheless this same model can be modified by optimizing for 6-DoF motion for each pixel in the keyframe aided by a *learned as-rigid-as-possible prior* on all the pixels in the keyframe. This will pave the way towards more reliable motion capture using only a single moving monocular camera in unconstrained settings.

Learning to Reconstruct and Reconstructing to Learn: For a purely vision-based system with no access to dedicated depth sensors, it will be fruitful to use our proposed unified model for “just-in-time” reconstruction (which more accurately models the ground truth depth maps) to provide feedback to the neural networks that parameterize them *where we are most confident at*, in such a way that the neural network predictions can be improved. For example in Chapter 4, Section 4.4.2 we saw that depth predictions improved features for matching, which in turn improved depth predictions, and in Chapter 3, Section 3.5.3 we alluded to how unsupervised depth estimation can serve as a weak training signal for learning surface normals which in turn can potentially improve depth predictions. Furthermore semantic segmentation can be improved by enforcing semantic label consistency across views based on the recovered geometry, even in the absence of semantic label groundtruth. Improved semantic mapping, localization, and unsupervised learning methods will pave the way for purely vision-based systems to achieve greater and continuous model improvement (in terms of both geometric and semantic scene understanding) through life-long operation and learning.

Bibliography

M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam-learning a compact, optimisable representation for dense visual slam. *arXiv preprint arXiv:1804.00874*, 2018.

C. Cadena, A. R. Dick, and I. D. Reid. Multi-modal auto-encoders as joint estimators for robotics scene understanding.

D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

A. Kundu, Y. Li, F. Dellaert, F. Li, and J. Rehg. Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In *Computer Vision – ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pages 703–718. 2014.

G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1925–1934, 2017.

J. McCormac, A. Handa, A. Davison, and S. Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4628–4635. IEEE, 2017.

T. T. Pham, I. Reid, Y. Latif, and S. Gould. Hierarchical higher-order regression forest fields: An application to 3d indoor scene labelling. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2246–2254, 2015.

V. Prisacariu, O. Kahler, M. Cheng, C. Ren, J. Valentin, P. Torr, I. Reid, and D. Murray. A Framework for the Volumetric Integration of Depth Images. *ArXiv e-prints*, 2014.

N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. June 2015.

C. Tang and P. Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.