Sparse and Dense Visual SLAM with Single-Image Depth Prediction

by

Shing Yan Loo

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

In this thesis, we investigate the use of single-image depth prediction from convolutional neural networks (CNNs) in sparse and dense monocular visual simultaneous localization and mapping (SLAM) problems. Mainly, we are interested in solving three problems: (1) data association, (2) dense mapping, and (3) long-term adaptation. Hence, we divide the thesis into three parts to discuss the contributions to solving the problems mentioned above.

To improve the robustness of data association in visual SLAM, our first proposal extends the state-of-the-art semi-direct visual SLAM algorithm using single-image depth prediction to improve the reliability of feature matching. We propose to use the additional depth information to initialize new features with a small uncertainty centred at the predicted depth. By reducing depth uncertainty, feature correspondence can be identified in a reduced search range along the epipolar line, resulting in fast convergence of the feature depth and improved mapping performance. With the improved mapping performance, our method outperforms the state-of-the-art visual SLAM algorithms in camera tracking error.

To recover a dense structure, we densify the semi-dense structure of the scene recovered from the state-of-the-art direct SLAM algorithm, LSD-SLAM. To this end, our second proposal exploits the local depth gradient consistency from single-image relative depth prediction as a spatial regularizer to densify the semi-dense depth maps. In addition, we propose an adaptive filtering scheme that incorporates the depth and pixel intensity of a local window to

reduce the noise of the semi-dense structure, which allows for a substantial gain in densification accuracy. The optimized semi-dense and densified structures, in turn, are being used to refine the pose-graph to refine the pose estimation. Experimental results show that our dense reconstruction accuracy outperforms the state-of-the-art methods by a large margin.

Nevertheless, single-image depth prediction from CNNs tends to give accurate depth estimations on images similar to that of the training images. Therefore, to improve the generality of single-image depth prediction used in visual SLAM, our third proposal introduces a long-term adaptation framework, which supports online fine-tuning of a depth prediction CNN to improve its accuracy while leveraging improved quality of depth prediction to optimize the structure and camera pose estimation globally. Particularly, we propose a novel online adaptation method in which the fine-tuning is enhanced with regularization to retain the previously learned knowledge while the CNN is continually trained. We demonstrate the use of fine-tuned depth prediction for map point culling before running global photometric BA, resulting in a more accurate map reconstruction than running global photometric BA on all map points.

Preface

All of the published work and research conducted for this thesis are the results of a collaboration between the University of Alberta and Universiti Putra Malaysia under the *Dual Ph.D. Programme*, led by Prof. Hong Zhang at the University of Alberta, with Prof. Sai Hong Tang and Prof. Syamsiah Mashohor being the supervisors at the Universiti Putra Malaysia.

The thesis chapters are based on the following publications.

- Chapter 3: Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. "CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction." In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5218-5223. IEEE, 2019.

- Chapter 4: Shing Yan Loo, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. "DeepRelativeFusion: Dense Monocular SLAM using Single-Image Relative Depth Prediction." In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6641-6648. IEEE, 2021.

- Chapter 5: Shing Yan Loo, Moein Shakeri, Sai Hong Tang, Syamsiah Mashohor and Hong Zhang. "Online Mutual Adaptation of Deep Depth Prediction and Visual SLAM." *Under review*, 2022.

Shing Yan Loo was responsible for the proposed methodology, implementation and evaluation of the solutions. Hong Zhang, Sai Hong Tang, and Syamsiah Mashohor provided suggestions and proofread the papers. Particular acknowledgement goes to Ali Jahani Amiri and Seyed Moein Shakeri for evaluating the state-of-the-art methods and helping with the experiments.

To my wife

For the unwavering support and love.

Acknowledgements

Prof. Hong Zhang has given me a once-in-a-lifetime opportunity to study in Canada and taken me under his wing. I want to thank him for providing a safe place for me to learn-from-failures, which helps me become a better researcher and, more importantly, a better person.

Many thanks go to my co-supervisors, Prof. Sai Hong Tang and Prof. Syamsiah Mashohor, for the kind support academically and mentally during my time at the Universiti Putra Malaysia. Also, I would like to express my gratitude to my supervisory and thesis committee members, Prof. Nilanjan Ray, Prof. Matt Taylor, Prof. Ron Kube, Prof. Li Cheng and Prof. Ping Tan, for participating and asking thought-provoking questions in the progress meetings, seminars and my thesis defence.

Of course, this journey would not be complete without my wonderful lab mates. I am grateful to have Ali, Sean, Weinan, Xuebin, Nazmus, Moein and Islam for their incredible help in experiments and insightful discussions. I appreciate the technical support from Martin and Ehsan for keeping the lab server machine up and running. Last but not least, I would like to thank Sara for offering her help when I did not sleep for two days working on a demo.

Lastly, I would like to express my gratitude to my wife, Renzi, and my family for being a great support system in my life.

List of Publications

(*In chronological order*)

1. Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. "CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction." In 2019 International Conference on Robotics and Automation (ICRA), pp. 5218-5223. IEEE, 2019.

2. Ali Jahani Amiri, Shing Yan Loo, and Hong Zhang. "Semi-supervised monocular depth estimation with left-right consistency using deep neural network." In 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 602-607. IEEE, 2019.

3. Moein Shakeri, Shing Yan Loo, Hong Zhang, and Kangkang Hu. "Polarimetric Monocular Dense Mapping Using Relative Deep Depth Prior." In IEEE Robotics and Automation Letters (RAL), 6(3), pp. 4512-4519, IEEE, 2021.

4. Weinan Chen, Lei Zhu, Shing Yan Loo, Jian Kun, Chaoqun Wang, Max Qinghu Meng, and Hong Zhang. "Robustness Improvement of Using Pre-trained Network in Visual Odometry for On-road Driving." IEEE Transactions on Vehicular Technology (TVT), IEEE, 2021.

5. Shing Yan Loo, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. "DeepRelativeFusion: Dense Monocular SLAM using Single-Image Relative Depth Prediction." In 2021 International Conference on Intelligent Robots and Systems (IROS), pp. 6641-6648. IEEE, 2021.

6. Shing Yan Loo, Moein Shakeri, Sai Hong Tang, Syamsiah Mashohor, and Hong Zhang. "Online Mutual Adaptation of Deep Depth Prediction and Visual SLAM." *under review*, 2021.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter begins with the motivation (Section 1.1) and background (Section 1.2) of the thesis. Next, we outline the problems of the study in Section 1.3 and define the main contributions in Section 1.4. Then, we delineate the structure of the thesis in Section 1.5.

1.1 Motivation

Vision is a powerful sensory modality to understand our surroundings. Notably, as humans, we depend on vision predominantly to perceive our surroundings and navigate from point A to point B. However, it is still a challenging problem for robots to reconstruct the environment and localize within the environment as an egocentric agent.

The core of the problem is to recover the 3D structure from which the image pixels are formed and the camera locations at which the images are taken [1], [2], which is particularly challenging due to the loss of depth information in the image capturing process, as illustrated in Figure 1.1. Solving the problem, also known as the visual simultaneous localization and mapping (SLAM) problem [3], can benefit numerous applications: autonomous navigation systems and virtual and augmented realities. For instance, autonomous navigation vehicles require constant localization with respect to the map while expanding the map with the *newly* observed visual features; similarly, augmented reality expands the map with artificial objects, which requires faithful reconstruction of the surroundings and reliable tracking of the camera locations. With the

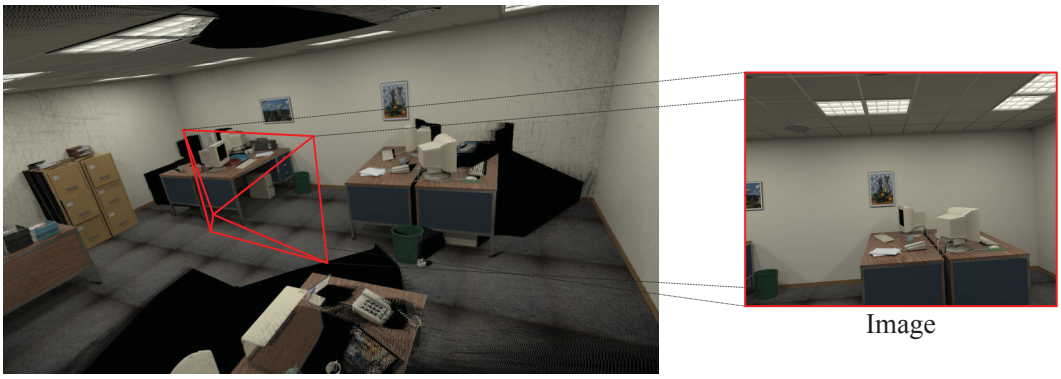

Figure 1.1: Perception of the 3D world through a camera. The image shown on the right is formed by capturing rays of light in front of the camera location in the 3D world. An illustration generated using the ICL-NUIM dataset [5].

recent development of self-driving cars and Metaverse [4] technologies, there has not been a more important time to contribute to solving visual SLAM problems.

On the other hand, deep learning has gained tremendous progress in terms of providing useful solution to technical challenges in solving the SLAM problem. Specifically, convolutional neural networks (CNN) have been able to predict surface normals [6], optical flow [7], keypoints [8] and depth maps from image input, information that can be exploited for improving SLAM performance [9]–[15]. In particular, single-image depth prediction is useful in the sense that it helps mitigate the *mapping* part of the SLAM problem, as the predicted depth information can be used to reconstruct the scene (see Figure 1.2). With the increasing amount of training data from real-world scenarios (e.g., KITTI [16], Oxford Robotcar [17], NYUv2 [18] and ScanNet [19]) and sophisticated simulators (e.g., Microsoft AirSim [20], Carla Simulator [21] and Nvidia Omniverse$^{\text{TM}}$ Isaac Sim [22]), we have seen tremendous opportunities to apply deep depth prediction for improving SLAM performance.

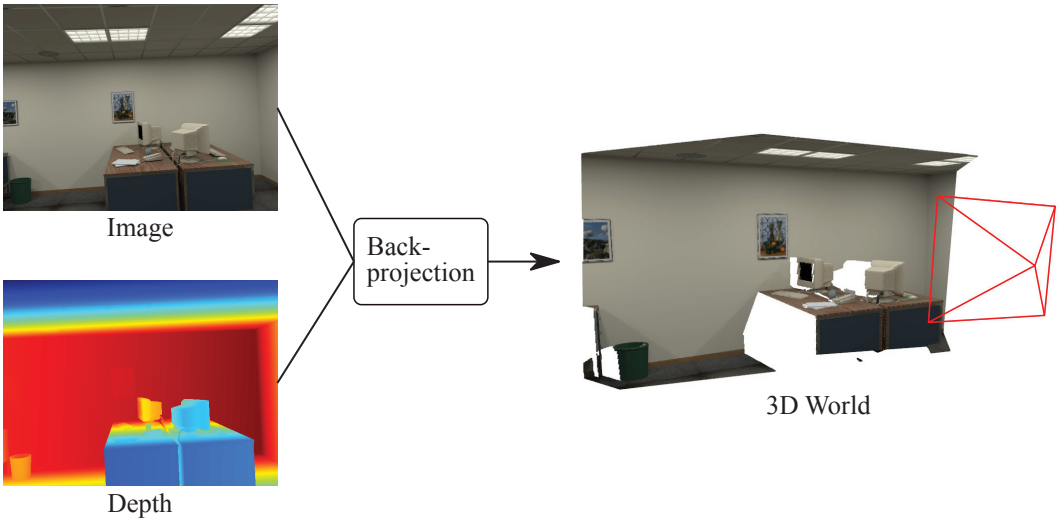

Figure 1.2: With a depth map, where blue is near and red is far, we can back-project the image pixels to reconstruct the 3D world. An illustration generated using the ICL-NUIM dataset [5].

1.2 Background

This section provides a general overview of the visual SLAM algorithms and single-image depth prediction. These are two research topics that serve as the basis of the research described in this thesis.

1.2.1 Visual SLAM

Solving SLAM requires the formulation of the SLAM problem. In general, there are two approaches to solving SLAM [1], [2], [23], [24]: filtering and smoothing. Filtering-based approaches (e.g., extended Kalman filter (EKF) [25], [26], Rao-Blackwellized particle filter [27], [28], and MonoSLAM [26]) solve the SLAM problem using Bayesian inference, i.e., assuming a Markov chain[1] [29], the current state (the current robot pose and the map) of the robot is sequentially updated based on the sensory observation and the previous state; alternatively, smoothing-based approaches (e.g., smoothing and mapping (SAM) [30], [31] and graph-based SLAM [32], [33]) optimize over all the robot poses and

[1]In the context of *online* state estimation, one of the Markov chain properties is that the current state estimation is dependent on the previous state.

the map. The latter is considered to be a *modern* approach for tackling the SLAM problem [2].

A typical design of a modern SLAM algorithm consists of a front-end and a back-end (see Figure 1.4). For a visual SLAM algorithm, the front-end extracts and matches useful features from the images captured by a camera, computes the relative motion between the camera poses[2] and constructs a graph in which a set of nodes (the camera poses and position of map points[3]) and edges (the constraints between the nodes) are contained. Then, the back-end performs *maximum a posteriori* (MAP) estimation to optimize the graph. In the following, we give a brief introduction to the front-end and back-end of a visual SLAM algorithm, and then compare the state-of-the-art visual SLAM algorithms.

Visual SLAM front-end

The primary purpose of the front-end is to establish constraints a graph. In this thesis, we consider two main constraint types, relative motion constraint and feature matching constraint. Relative motion constraint determines the relative transformation between two camera poses. On the other hand, feature matching constraint contains a measurement between a new visual features and an existing feature, e.g., matching a corner of a table in two images.

We can impose two types of relative motion constraints, namely odometry and loop constraints. Odometry constraint is a short-term constraint that can be established by defining pairwise camera transformations while the camera moves. Loop constraint, on the contrary, is a long-term constraint that defines the relative transformation between two distant camera poses, which requires loop closure detection [34]–[36] to determine if the camera revisits a previously mapped area.

Moreover, matching features can be done directly or indirectly. In a direct formulation, matching operates directly on the raw pixel level to identify feature correspondences [37], [38]. Therefore, any arbitrary pixels (e.g., corners

[2]As a camera is rigidly attached to a robot, recovering the camera poses is equivalent to recovering the robot poses.

[3]We review the concept of camera pose and map point in Section 2.3.2.

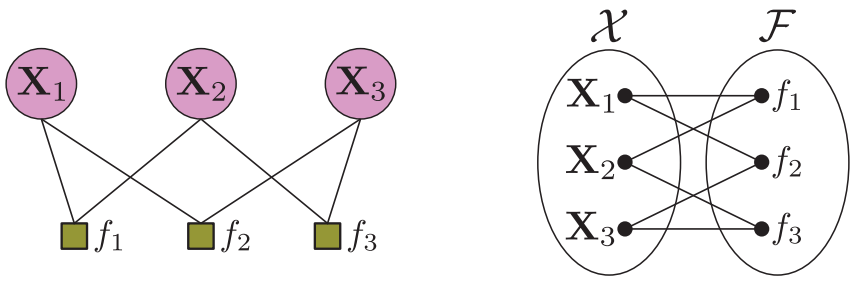

Figure 1.3: A factor graph consisting of three variable vertices and two factor vertices with edges connecting a variable node to a factor node such that the variable vertices $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$ and the factor vertices $\{f_1, f_2, f_3\}$ belong to two disjoint and independent sets $\mathcal{X}$ and $\mathcal{F}$, respectively.

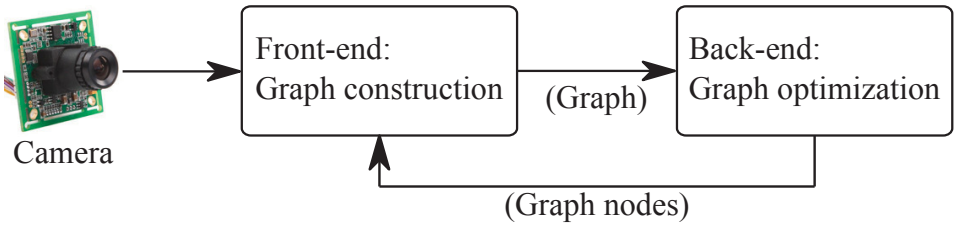

Figure 1.4: Front-end and back-end design in a visual SLAM algorithm.

or edges) can be extracted and matched, assuming photometric consistency. However, the photometric consistency assumption does not always hold in practice. One way to maximize the photometric consistency is to optimize the affine brightness transformation across images by shifting and scaling the pixel intensity globally [13], [38]. On the other hand, an indirect formulation requires feature extraction, feature description, and matching of feature descriptors [33], [39], [40]. The descriptor matching performance mainly depends on maximizing descriptor distance between two non-matching features while minimizing descriptor distance between two matching features. As laptops and embedded computers are getting more powerful, there are research efforts proposing the use of hardware acceleration and parallelism for CNN feature descriptors [41], [42]), which can be expensive to compute. Once the features are matched, new nodes and edges that define the constraints in the graph can be formed.

We can use a factor graph to solve the visual SLAM problem [32], which is a bipartite graph $G = (\mathcal{X}, \mathcal{F}, \mathcal{E})$ consisting the variable vertices $\mathcal{X}$, factor vertices $\mathcal{F}$, and edges $\mathcal{E}$ of the graph. Without getting into details of solving the visual SLAM problem[4], suppose the factor graph consists of three SLAM variables $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$, three factor vertices $\mathcal{F} = \{f_1, f_2, f_3\}$ and six edges $\mathcal{E} = \{\epsilon_{11}, \epsilon_{12}, \epsilon_{21}, \epsilon_{23}, \epsilon_{32}, \epsilon_{33}\}$, as illustrated in Figure 1.3. We can perform a factorization of a function f into a product of sub-functions over the *local* variables [43]:

$$f(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3) = f_1(\mathbf{X}_1, \mathbf{X}_2) f_2(\mathbf{X}_1, \mathbf{X}_3) f_3(\mathbf{X}_2, \mathbf{X}_3), \tag{1.1}$$

where each sub-function corresponds to a constraint.

Visual SLAM back-end

The purpose of the back-end is to optimize the graph constructed by the front-end. Optimization of the graph can be formulated as a MAP problem, which seeks to estimate the variable $\mathcal{X}$ that maximizes the posterior $p(\mathcal{X}|\mathcal{Z})$

[4]More details can be found in Appendix-B

according to the Bayes theorem:

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} p(\mathcal{X}|\mathcal{Z}) = \arg\max_{\mathcal{X}} p(\mathcal{Z}|\mathcal{X})p(\mathcal{X}), \tag{1.2}$$

where $\mathcal{Z}$ are the measurements defined by the constraints from the front-end. Assuming that the measurements are independent, we can rewrite Equation 1.2 as:

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} p(\mathcal{X}) \prod_{k=1}^{m} p(\mathbf{z}_k|\mathcal{X}_k), \tag{1.3}$$

where there are m measurements and that the measurement $\mathbf{z}_k$ only relates to $\mathcal{X}_k \subseteq \mathcal{X}$. Further assuming that the measurement noise is Gaussian (a zero-mean noise ϵ_k with information matrix[5] $\mathbf{Q}_k$). The measurement likelihood and the prior becomes:

$$p(z_k|\mathcal{X}_k) \propto \exp(-\frac{1}{2} \left\| h(\mathcal{X}_k) - \mathbf{z}_k \right\|^2_{\mathbf{Q}_k}) \tag{1.4}$$

and

$$p(\mathcal{X}) \propto \exp(-\frac{1}{2} \left\| h(\mathcal{X}_0) - \mathbf{z}_0 \right\|^2_{\mathbf{Q}}), \tag{1.5}$$

respectively. $h(\cdot)$ is a function that is known as the measurement model, $\|\mathbf{e}\|^2_{\mathbf{Q}_k}$ is a shorthand for $\mathbf{e}^T\mathbf{Q}_k\mathbf{e}$, and $\mathbf{z}_0$ and $\mathbf{Q}_0$ are the prior measurement and information matrix.

Since maximizing the posterior is equivalent to minimizing the negative log-likelihood, we can re-write Equation 1.3 in an explicit form:

$$\begin{aligned}
\mathcal{X}^* &= \arg\min_{\mathcal{X}} -\log\left(p(\mathcal{X}) \prod_{k=1}^{m} p(\mathbf{z}_k|\mathcal{X}_k)\right) \\
&= \arg\min_{\mathcal{X}} -\log \prod_{k=0}^{m} p(\mathbf{z}_k|\mathcal{X}_k) \\
&= \arg\min_{\mathcal{X}} \sum_{k=0}^{m} \left\| h(\mathcal{X}_k) - \mathbf{z}_k \right\|^2_{\mathbf{Q}_k}.
\end{aligned} \tag{1.6}$$

Therefore, minimizing the negative log-likelihood becomes a least-squares problem, which can be solved iteratively via Gauss-Newton or Levenberg-Marquardt methods. To deal with outliers in the measurements, we may substitute the squared Mahalanobis norm with a more robust norm (Huber or Tukey) [44].

[5]Information matrix is the inverse of the covariance matrix.

Table 1.1: A comparison between modern visual SLAM algorithms in chronological order.

	Direct/indirect	Sparse/dense	Loop closure detection	Open-source
PTAM [48]	Indirect	Sparse	✗	✓
DTAM [49]	Direct	Dense	✗	✓
LSD-SLAM [37]	Direct	Semi-dense	✓	✓
SVO [50], [51]	Semi-direct	Sparse	✗	✓
DSO [38]	Direct	Sparse	✗	✓
LDSO [52]	Direct	Sparse	✗	✓
VITAMIN-E [53]	Direct	Dense	✗	✗
OpenVSLAM [54]	Indirect	Sparse	✓	✓[1]
UcoSLAM [55]	Indirect	Sparse	✓	✓
DSM [56]	Direct	Sparse	✗	✓
ORB-SLAM [33], [57], [58]	Indirect	Sparse	✓	✓

[1]Archived; no official support for future releases.

More details on the visual SLAM graph optimization can be found in Appendix B.

State-of-the-art visual SLAM algorithms

Table 1.1 compares the state-of-the-art modern visual SLAM algorithms. In general, there is no one dominant solution for solving all SLAM problems [45]; rather, the SLAM performance strongly depends on the scene characteristics and camera types. For example, direct methods can use all the pixels in the images, but are prone to fail under the rolling shutter effect, illumination changes, large interframe distance, etc. [38]. On the other hand, indirect methods tend to fail in texture-poor scenes in which limited feature descriptors can be calculated and matched [45]. To further improve SLAM performance, loop closure detection [34]–[36] is performed in the front-end to determine if the most *recent* image is similar to one of the previously captured images, and an additional loop constraint is created upon detection. Failing to detect a loop might result in the "infinite corridor" problem, as the tracked camera motion could drift over time (see Figure 1.5). Note that certain literature uses loop closure as a differentiator to highlight the difference between visual SLAM and visual odometry (VO) [46], where visual SLAM produces a globally consistent estimation of camera trajectory and map [3], [47]. Instead, we refer to both visual SLAM and VO as visual SLAM throughout the thesis.

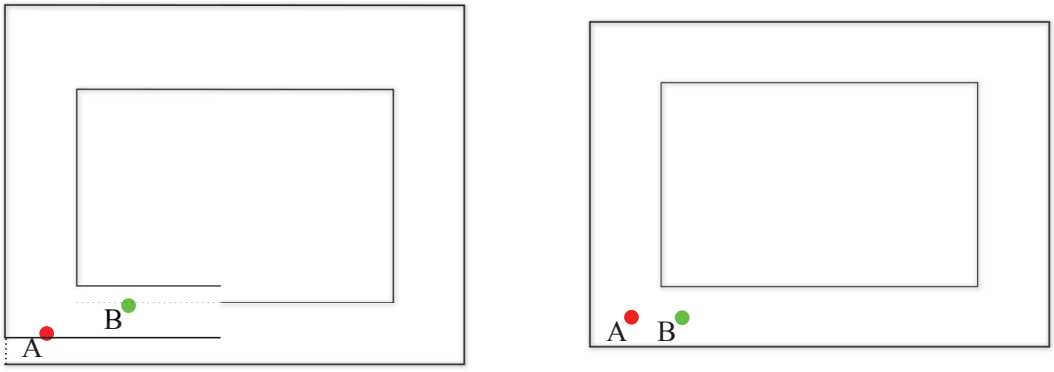

Figure 1.5: Illustration of the "infinite corridor" problem where a robot drives around a rectangular hallway, starting at point A, travelling counterclockwise, and ending at point B. Without loop closure (left), the accumulated drift causes inconsistency between the mapped trajectory and the actual map, whereas, with loop closure (right), the mapped trajectory is corrected to match the actual rectangular hallway. Therefore, without loop closure, the robot perpetually "discovers" new structures in the environment, hence the "infinite corridor" problem.

1.2.2 Single-image depth prediction

Single-image depth prediction is the task of predicting a depth map from an image. Without leveraging multiple viewpoints, traditional (*shallow*) methods are designed to operate on constrained assumptions, e.g., shape-from-vanishing-points [59], shape-from-defocus [60], and shape-from-shading [61]. In recent years, single-image depth prediction using deep neural networks has been an active research topic [62], with the advantages of being able to learn from training data and good depth prediction accuracy. Specifically, CNNs have been widely used for solving image-based tasks and have been shown to predict depth with state-of-the-art accuracy[6,7]. Though we are well aware that dense vision transformers [63], a new architecture for solving image-based tasks, could potentially replace or combine with CNNs for greater learning capabilities, it can in principle be adapted for use by the methods presented in this thesis. For now, we will only present a general overview of CNNs and

[6] `http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_prediction`

[7] `https://paperswithcode.com/task/depth-estimation`

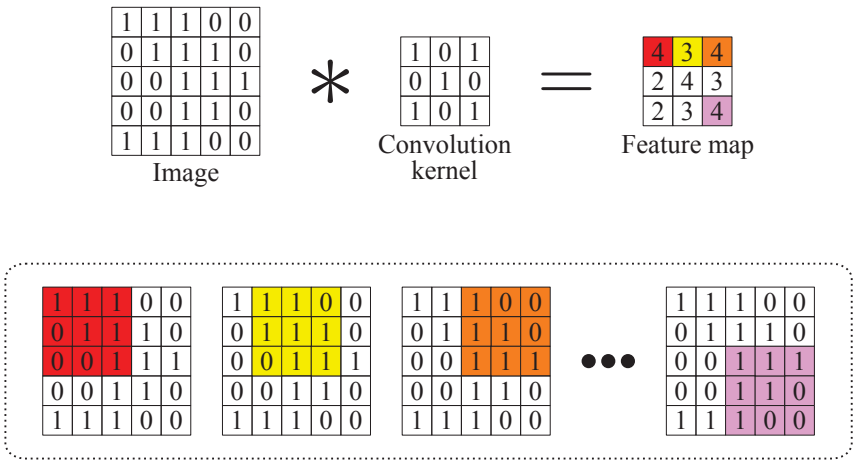

Figure 1.6: A demonstration of 2D convolution operation for generating a convolved feature. The coloured *pixel* on top is generated by the weighted combination of the corresponding image region in the bottom row with the weights in the convolution kernel. Note that the bias parameter is not included in this illustration.

types of single-image depth predictions in the following.

Convolutional neural networks (CNNs)

A convolutional neural network mainly consists of convolutional layers to extract useful visual features in the training process. Each convolutional layer generates a set of feature maps, and relies on a convolution operation using a trainable convolution kernel (see Figure 1.6 for a minimal example). After obtaining the feature maps, we may apply an activation function to obtain the nonlinearity mapping between the input and output:

$$\mathbf{H}^k = f(\mathbf{W}^k * \mathbf{X} + \mathbf{b}), \tag{1.7}$$

where $\mathbf{H}^k$ is the k-th feature map, $\mathbf{W}^k$ the k-th convolution kernel, $\mathbf{X}$ the input map, $\mathbf{b}$ the bias, and $f(\cdot)$ the activation function.

Besides convolutional layers, pooling layers have also been used to reduce the dimension of the feature maps in a CNN, allowing for multi-scale feature representation and interaction. Figure 1.7 demonstrates two commonly used pooling layers: max pooling and average pooling.

Effective combinations of convolutional layers, activation functions, pooling

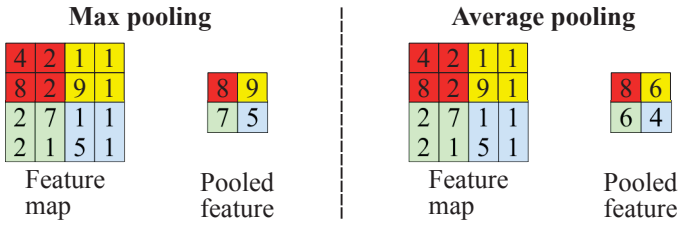

Figure 1.7: A demonstration of max pooling (left) and average pooling (right)

layers and fully connected layers[8] together with a dataset and proper training enable a CNN to map an image input to a depth map output [62]. We will discuss the types of single-image depth prediction using a CNN in the following sub-section and the training methods in Section 2.4.

Types of single-image depth predictions

There are two types of single-image depth predictions: absolute depth prediction and relative depth prediction. In absolute depth prediction, a CNN is trained to predict the metric depth maps from single images [6], [64]–[66]. Because of the CNN prediction range, the CNN training is commonly limited to one scene type, e.g., indoor or outdoor[9]. On the other hand, relative depth prediction is concerned with estimating the distance of one space point with respect to the others, i.e., their depth order, rather than the absolute depth.

Training a CNN to predict absolute depth is popularized by the unsupervised (sometimes referred to as self-supervised) learning of depth by minimizing the photometric reconstruction errors of the corresponding left/right images from the predicted disparity maps, whereby a static stereo camera setup captures left-right image pairs[10] [65], [67]. The technique of reconstructing the corresponding image is also known as novel view synthesis and has been applied to reconstruct nearby views in a monocular sequence to train depth and pose prediction CNN's end-to-end [68]–[70]. Even with the absence

[8] As its name suggests, each neuron in the fully connected layer is connected to all neurons of the previous layer (and we need to flatten the feature maps if the previous layer is not in the form of a vector.)

[9] Two commonly used benchmarks are the KITTI Eigen split [16] and NYUv2 dataset [18].

[10] The relationship between disparity and depth is given by $z = \frac{f \cdot b}{d}$, where z is the depth, f the camera focal length, b the stereo baseline, and d the disparity.

of pose prediction CNNs, geometric cues (camera poses and sparse structure) can be provided externally (e.g., SLAM, depth sensors, IMUs) to synthesize novel views and provide sparse geometric supervision to train depth prediction CNNs [12], [71], [72]. Since then, remarkable progress has been made to advance the state-of-the-art, ranging from novel training methods (e.g., semi-supervised with sparse ground truth [64], [73], 3D consistency, normal/planar constraint [6], [66], [74], and flow field consistency [75]) to new CNN architectures (e.g., generative adverserial networks [76], 3D packing/unpacking convolutional blocks [77], dense prediction transformers [63] and discretized depth output bins [78], [79]).

On the other hand, early work on relative depth prediction learns from ordinal depth annotations (closer/farther relationship between two points), which contain relatively accurate sparse depth relationships covering a wide range of scene types (e.g., mixing indoor and outdoor scenes in a combined training dataset) [80], [81]. The training results demonstrate accurate ordinal depth prediction quantitatively on different datasets and qualitatively on unconstrained photos taken from the internet, albeit in the absence of absolute depth values. To train on more extensive and diverse datasets, Lasinger et al. propose to train a relative depth prediction CNN, named MiDaS [82], using a scale- and shift-invariant loss, which handles unknown depth scale and global shift factors in different datasets. Yin et al. further disentangle the scaling and translation (surface normals) by combining scale- and shift-invariant loss with virtual normal loss to predict affine-invariant depth [83]. Furthermore, due to scale, shift and surface normals disentanglements in the training loss, CNNs can be trained to predict depth maps on a large variety of scene types without the metric scale. To differentiate between absolute and relative depth prediction, we from now on refer to single-image absolute depth prediction as single-image depth prediction, unless a comparison between the two is taking place.

1.3 Problem Statements

This thesis seeks to investigate the benefits and limitations of using single-image absolute and relative depth predictions to solve monocular visual SLAM problems. Formally, this thesis attempts to address three fundamental questions.

1. **How can we improve feature correspondence in sparse visual SLAM?**

   Mapping involves the reconstruction of a 3D structure from an image sequence, a critical step to ensure the continuation of camera tracking in the subsequent frames. To reconstruct the 3D structure, triangulation is performed by the tracked camera poses and the matched features between images, also known as the feature correspondences. Therefore, the mapping problem becomes the problem of identifying accurate feature correspondences. Feature correspondences can be done through the matching of feature descriptors (indirect formulation) [33], [39], [40] and patterned image patches (direct formulation) [37], [38], [50]. In this thesis, we opt for the latter for identifying feature correspondences for two reasons. First, calculating photometric errors between two image patches is faster than extracting and matching feature descriptors. Second, additional depth information can be conveniently incorporated into the feature matching step; given known camera poses of the images, we can exploit epipolar geometry by identifying the corresponding feature with the lowest photometric matching cost along the epipolar line. However, as the distance between two cameras grows, photometric matching becomes an issue as the number of local minima also increases in the feature correspondence search (see Figure 1.8). With single-image depth prediction, the search range along the epipolar line can be narrowed down, which helps disambiguate a true feature match from a false match.

2. **How can we densify a semi-dense map from visual SLAM?**

   Dense reconstruction is especially challenging in the texture-poor im-

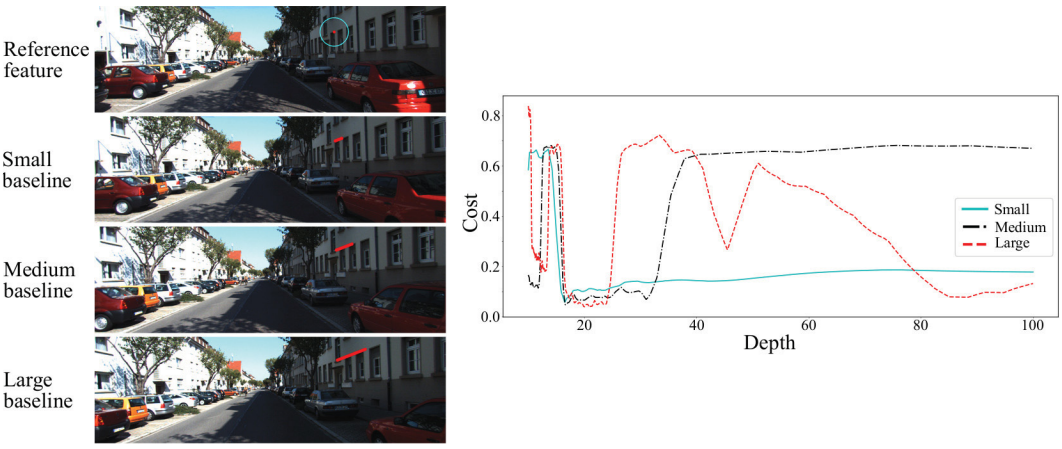

Figure 1.8: A comparison of photometric errors of various stereo baselines. From top to bottom: A reference feature in a reference frame (the circled red dot), three consecutive images in a sequence with increasing motion stereo baselines, and the photometric matching errors at varying depth. The red line is the re-projected pixel locations along the epipolar line. A small baseline gives a distinct minimum at around the depth of 15 m, and an increasing stereo baseline results in an increase in local minima in matching cost.

age regions due to the lack of minima in photometric matching (see Figure 1.9). Without distinct local minima in photometric matching, dense reconstruction cannot be performed using conventional triangulation [38], [50]. Nevertheless, traditional dense monocular SLAM algorithms rely on regularization techniques to enable dense map reconstruction, such as piecewise planar [84], [85], Manhattan assumption [86], smoothness [49], and plane sweeping [87], [88]. Recently deep learning-based regularization techniques have been proposed that use depth [15], [89] or surface normals [9] information estimated by CNNs as a regularizer to create a dense map. Motivated by the high accuracy of single-image relative depth prediction across novel domains, we study the problem of optimizing and densifying a semi-dense structure by exploiting the local depth gradient consistency from single-image relative depth prediction, and using the optimized structure to refine the pose estimation.

3. **How can we adapt a depth prediction network online to a novel**

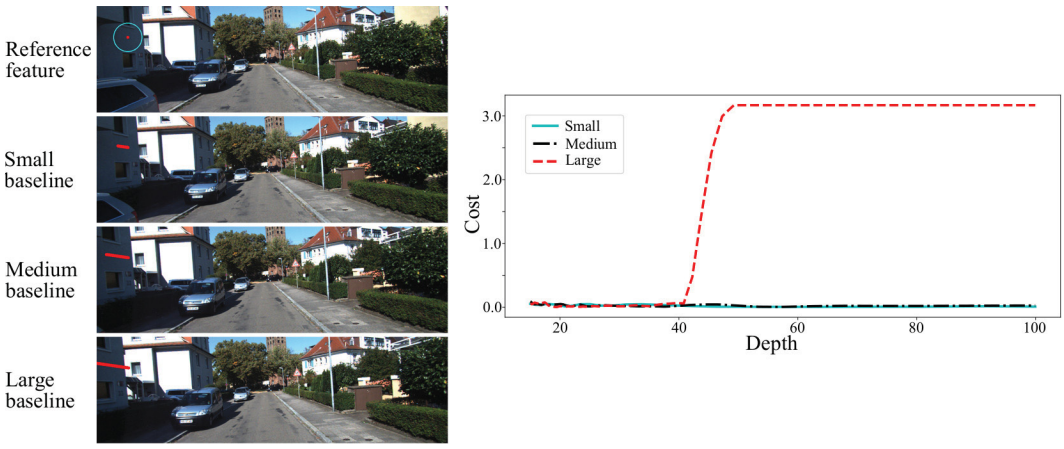

Figure 1.9: A comparison of photometric errors of various stereo baselines in textureless image regions. From top to bottom: A reference feature in a reference frame (the circled red dot), three consecutive images in a sequence with increasing motion stereo baselines, and the photometric matching errors at varying depth. The red line is the re-projected pixel locations along the epipolar line. Compared to matching a textured corner in the nearby images (see Figure 1.8), a textureless region cannot be localized in the nearby images using photometric error.

environment in order for us to use it to improve depth prediction and visual SLAM accuracy?

Single-image depth prediction needs to be accurate for it to be used to improve SLAM performance. However, single-image depth prediction suffers from the generalization problem, where the depth prediction accuracy cannot be generalized across the pre-trained domain. Figure 1.10 illustrates the generalization problem of single-image absolute depth prediction, where the depth prediction accuracy is not transferrable to a novel domain (e.g., trained in an indoor environment and used in an outdoor environment); similarly, Figure 1.11 shows that even single-image relative depth can be inconsistent in depth prediction accuracy in the operating environment. Evidently, the accurate depth prediction assumption does not hold in practical robotics applications. Therefore, we investigate the problem of visual SLAM with online adaptation by on-demand fine-tuning of a depth prediction CNN before integrating online

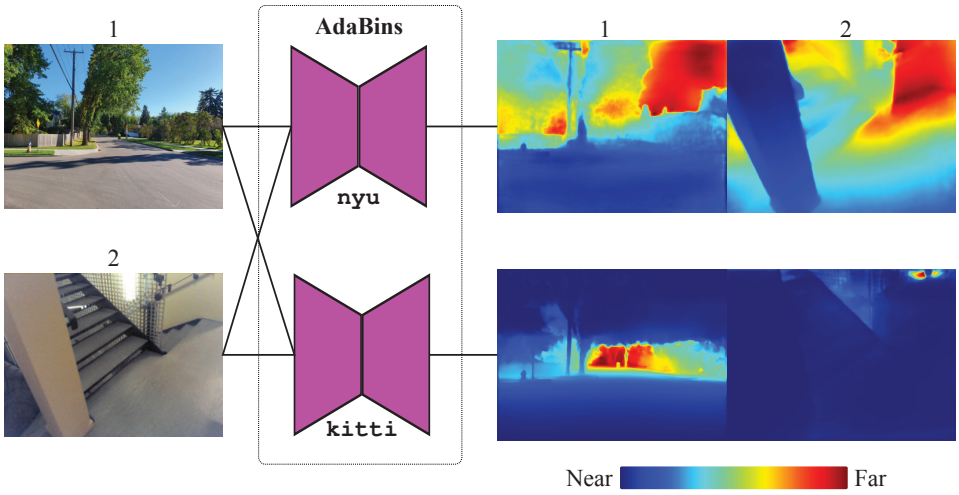

Figure 1.10: A comparison of the generalizability of two state-of-the-art depth prediction models by AdaBins [79], `nyu` and `kitti`, which have been trained on the (indoor) NYUv2 [18] and (outdoor) KITTI [16] datasets, respectively. We test the pre-trained models on two input images (1 and 2), and the predicted depth maps are shown on the right. Image 2 is taken from the 7 scenes dataset stairs sequence [90]. Note that the AdaBins `kitti` model is not able to predict accurate depth in the upper image region, due to the absence of ground truth depth in the training, and is typically ignored in the quantitative evaluation.

adapted depth prediction and using it to improve structure and motion estimation.

1.4 Contributions

In this thesis, we propose novel solutions for improving monocular SLAM performance using single-image absolute and relative depth predictions. Our main contributions are as follows.

- In Chapter 3, we use the predicted depth information from single-image depth prediction to improve the feature matching performance, which, in turn, improves the mapping and camera tracking accuracies. Assume that the camera motion can be recovered reliably. Corresponding features can be found along their respective epipolar lines. We propose using single-image depth prediction to improve the feature matching performance by limiting the scope of the epipolar line search for identifying the corresponding features.

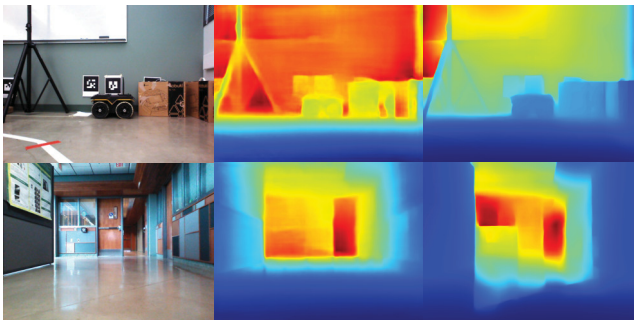

Figure 1.11: A comparison of single-image relative depth prediction by DiverseDepth [83] (middle) and MiDaS [82] (right).

- In Chapter 4, we use single-image relative depth prediction for densifying and optimizing a semi-dense structure from a monocular SLAM algorithm, and using the optimized structure to further improve the camera tracking accuracy. To densify the structure, we propose to exploit single-image relative depth prediction as a *smoothness* prior in a GPU-accelerated energy minimization framework to *fill in* the texture poor image regions. To improve the densification accuracy, we also introduce two additional enhancements, an adaptive filter to remove noisy semi-dense depth pixels and pose-graph refinement, to further improve densification and camera tracking accuracies.

- In Chapter 5, we propose an online learning framework that consists of two complementary processes: a SLAM algorithm that is used to generate keyframes to fine-tune the depth prediction and another algorithm that uses the online adapted depth to improve map quality. Once the potential noisy map points are removed, we perform global photometric bundle adjustment (BA) [91] to improve the overall SLAM performance. To improve the fine-tuning accuracy, we introduce regularization to mitigate *catastrophic forgetting* [92] in the online learning of sequential data.

1.5 Thesis Outline

The thesis is organized as follows. Chapter 2 presents the fundamentals of visual SLAM in robotics, computer vision and deep neural network training

necessary for solving the problems in the subsequent chapters. Chapter 3 studies the problem of improving the mapping performance in visual SLAM. Chapter 4 investigates the problem of densifying and optimizing of a semi-dense structure from visual SLAM and then using the optimized structure to improve motion estimation. Chapter 5 looks into the problem of online adaptation to improve SLAM on the fly through the fine-tuning of single-image depth prediction. Chapter 6 concludes the thesis contributions and provides some potential future directions.

Chapter 2

Preliminaries

This chapter is divided into four parts. In Section 2.1, we define the mathematical notation used in this thesis. Then, the next three sections address the primals of visual SLAM in robotics (Section 2.2), computer vision (Section 2.3) and learning of single-image depth prediction (Section 2.4). The goal is to provide a background knowledge required for the rest of the chapters.

2.1 Mathematical notation

We use upper case letters (e.g., X and M) to denote sets. In addition, we use bold upper case letters (e.g,, $\mathbf{T}$ and $\mathbf{R}$) and lower case letters (e.g., $\mathbf{a}$ and $\mathbf{u}$) for representing matrices and vectors, respectively. Lastly, we use function notations to express the access of image I, depth map D and variance map V values:

$$I : \Omega \to \mathbb{R}^+ \tag{2.1}$$

$$D : \Omega \to \mathbb{R}^+ \tag{2.2}$$

$$V : \Omega \to \mathbb{R}^+, \tag{2.3}$$

where $\Omega \subset \mathbb{R}^2$ is a set of valid pixel coordinates.

2.2 Visual SLAM in robotics

What is the visual SLAM problem? To illustrate the problem, suppose a mobile robot (e.g., wheeled, legged and flying robots) is moving around in an un-

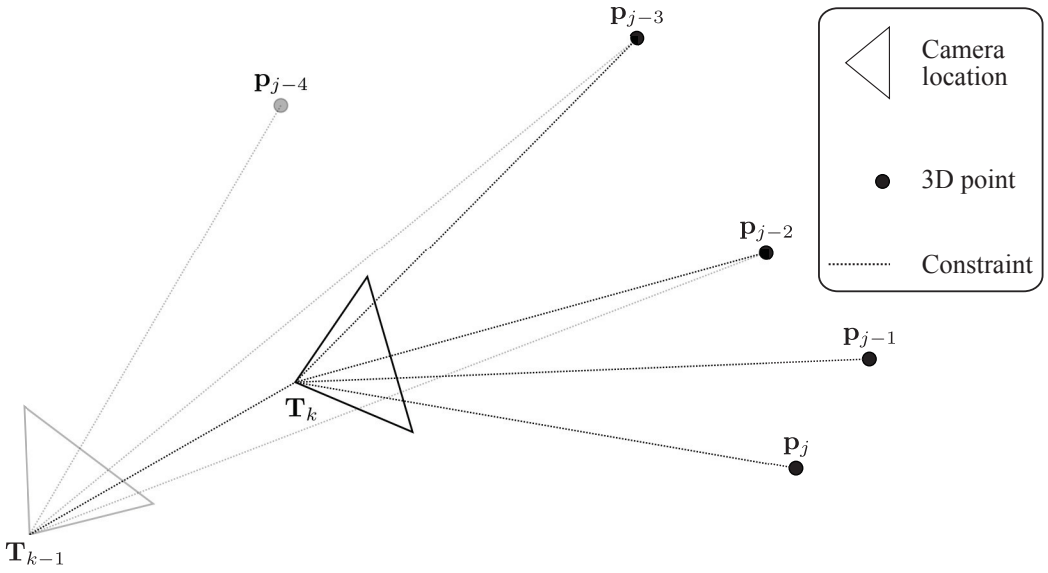

Figure 2.1: The visual SLAM problem. The new camera location is recovered using the existing 3D points as constraints, while new 3D points are also being initialized at the location.

known environment and incrementally build a map while determining its location. Formally, given an image stream captured by a camera attached to a mobile robot, the structure of the problem can be defined as the recovery of a set of camera locations $T = \{\mathbf{T}_1, \mathbf{T}_2 \ldots, \mathbf{T}_k\}$, local maps $M = \{M_1, M_2, \ldots, M_k\}$, and constraints $C = \{C_1, C_2, \ldots, C_k\}$ at successive discrete time steps. Then, we can solve for the optimal locations of the 3D points and cameras using least-squares.

However, considerable effort is required to solve the visual SLAM problem. For instance, having too many constraints can be computationally prohibitive in the back-end least-squares optimization. Conversely, having too little constraints may result in poor estimation of the 3D points and camera locations. Therefore, a visual SLAM algorithm will need to formulate an adequate set of constraints by sampling a subset of images from the image stream, known as the keyframes, and extracting *high quality* features in the keyframes for creating 3D points.

2.3 Projective geometry and 3D transformations

This section provides the fundamentals of computer vision and mathematical concepts used in this thesis. First, we describe the relationship between a 3D point and a 2D image pixel coordinates in Section 2.3.1. Then, we discuss the transformation of 3D points in Section 2.3.2.

2.3.1 Camera projection

An image is a projection of the 3D world to a 2D plane (see also Figure 1.1). To perform the projection, we use the pinhole camera model[1], which describes the relationship between the 3D points (also known as *map points* in the SLAM context) and the 2D image coordinates (see Figure 2.2). Using projection of the 3D point $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ to the 2D image coordinates $\mathbf{a} = \begin{bmatrix} u & v \end{bmatrix}^T$ as an example, the projection can be expressed as:

$$\mathbf{a} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f_x \cdot x}{z} + c_x \\ \frac{f_y \cdot y}{z} + c_y \end{bmatrix}, \tag{2.4}$$

where f_x, f_y, c_x and c_y are the focal lengths and principal point of the camera, collectively known as the camera intrinsics ($\mathbf{K}$), obtained through camera calibration [95]. A more general expression can be expressed using a projection function $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$:

$$\mathbf{a} = \pi(\mathbf{p}), \tag{2.5}$$

where

$$\hat{\mathbf{a}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \mathbf{K} \mathbf{p} \tag{2.6}$$

$$= \lambda \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.7}$$

$$= \lambda \begin{bmatrix} f_x \cdot x + c_x \cdot z \\ f_y \cdot y + c_y \cdot z \\ z \end{bmatrix} \tag{2.8}$$

[1] Omnidirectional camera models (e.g., fisheye and catadioptric) [93] have also been studied in solving the visual SLAM problem [54], [94].

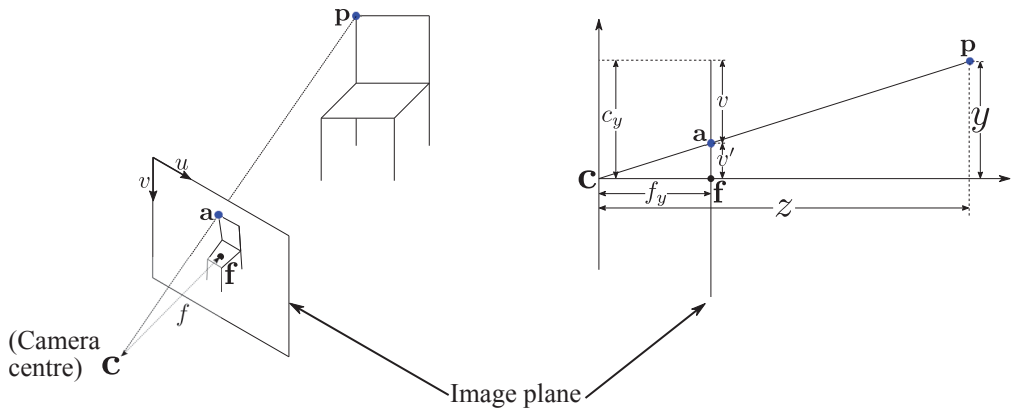

Figure 2.2: Left: Projection of a 3D object to the 2D image plane. Right: Projection of point p to the image plane, omitting the X-axis. See text for explanation of notation.

$\hat{\cdot}$ represents the homogeneous coordinates and λ is a normalization factor. By setting $\lambda = \frac{1}{z}$ and then removing the third row of the vectors in Equation 2.8, we arrive at the Equation 2.4. Similarly, the projection function can be inversed (i.e., $\pi^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$) to obtain a 3D point from a 2D image coordinates, given that the depth d is known:

$$\mathbf{p} = d\pi^{-1}(\mathbf{a}), \tag{2.9}$$

where

$$\pi^{-1}(\mathbf{a}) = \mathbf{K}^{-1}\hat{\mathbf{a}}. \tag{2.10}$$

Note that the images may be distorted and have to be undistorted before the pinhole model can be used (see Figure 2.3). For the experiments in the subsequent chapters, we assume that the images have been undistorted for the 3D-2D (π) and 2D-3D (π^{-1}) camera projections.

2.3.2 3D transformation

A map point may be observed in two or more images (or *frames* in the SLAM context). Therefore, we express the camera poses (i.e., the rotation $\mathbf{R}$ and translation $\mathbf{t}$ of the cameras with respect to a world frame) of the frames using rigid body transformation[2] to allow for the re-projection of a map point to

[2]Rigid body transformations preserve the angle and distance of the map points

Figure 2.3: Examples of distorted images. From left to right: image with no distortion, image with barrel distortion and image with pincushion distortion. Image taken from the ICL-NUIM dataset [5].

different frames. Moreover, through the re-projection of map points in multiple viewpoints, we can optimize the structure and camera motion. In the following, we introduce three essential concepts for solving the visual SLAM problem. First, we discuss the epipolar geometry, a constraint used for identifying feature correspondence between two images. Second, we introduce the definition of Lie groups and Lie algebras for transformation matrices. Third, we present the *linearization* (also known as the Jacobians) on the Lie group manifolds for structure and motion optimization.

Epipolar geometry

To map the scene captured by a camera, we need to triangulate 3D points from 2D image point correspondences between two images. To this end, let us consider two-view epipolar geometry shown in Figure 2.4. Assuming known relative rotation and translation between I_l and I_r, we can form an epipolar plane using the stereo baseline (the dashed line connecting I_l and I_r) and the vector defined by $\mathbf{a}$. With the epipolar plane, the corresponding point of $\mathbf{a}$ is restricted along the epipolar line (the blue line connecting the epipole $\mathbf{e}_r$ and $\mathbf{a}'$ in I_r). This is due to the depth ambiguity as image point a can be the projection of a 3D point along the ray defined by $\mathbf{a}$ (e.g., $\mathbf{p}$, $\mathbf{p}_1$ or $\mathbf{p}_2$). Once we get a match ($\mathbf{a} \leftrightarrow \mathbf{a}'$), a map point ($\mathbf{p}$) can be triangulated.

Lie groups and Lie algebras

Transformation matrices of the camera poses are 4×4 square matrices that belong to the Lie groups (denoted $\mathbf{T}$ and $\mathbf{S}$ in Table 2.1). The Lie groups used

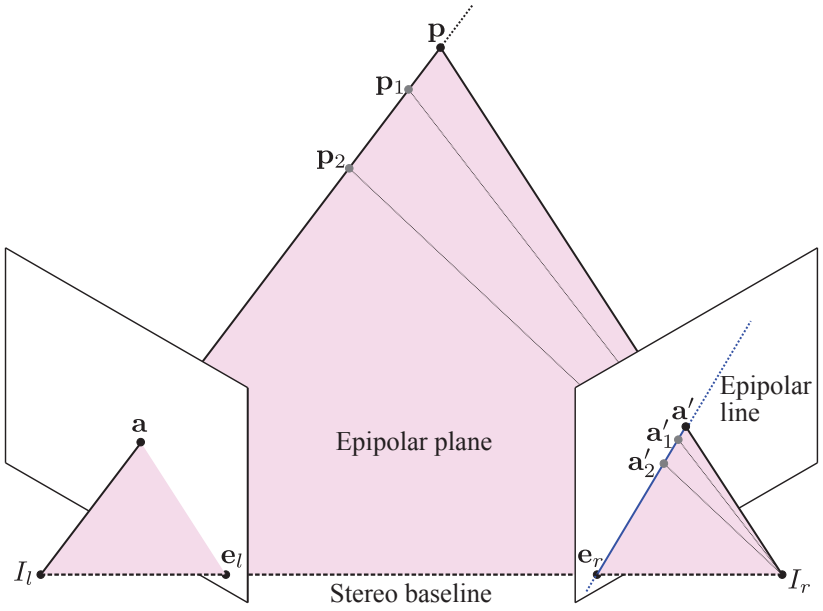

Figure 2.4: Given two images I_l and I_r and their relative rotation and translation, epipolar geometry dictates that the corresponding point of a lies along the epipolar line and that the epipolar plane contains the 3D point $\mathbf{p}$ and the re-projected image points $\mathbf{a}$ and $\mathbf{a}'$. The epipolar plane is determined by the vector defined by $\mathbf{a}$ and the stereo baseline.

in this thesis for solving the visual SLAM problem are the special orthogonal group SO(3), special Euclidean group SE(3) and 3D similarity transformations group Sim(3). A Lie group G is a differentiable manifold in which the group product ($\cdot : \mathrm{G} \to \mathrm{G}$) and inverse ($^{-1} : \mathrm{G} \to \mathrm{G}$) operations are smooth [96]. Note that Lie groups are not *closed* under the addition operation (i.e., adding two transformation matrices does not result in a transformation matrix), and therefore products of transformation matrices are used for concatenating camera poses; the inverse of a camera pose is simply the inverse of the transformation matrix. For example, given two camera poses with respect to the world frame, $\mathbf{T}_{1\to w}$ and $\mathbf{T}_{2\to w}$, a relative transformation can be obtained as follows:

$$\mathbf{T}_{1\to 2} = \mathbf{T}_{1\to w}\mathbf{T}_{2\to w}^{-1}, \tag{2.11}$$

where $\mathbf{T}_{2\to w}^{-1}$ is equivalent to $\mathbf{T}_{w\to 2}$.

A minimal representation of the Lie groups is defined by their corresponding Lie algebras shown in Table 2.2. The Lie algebras are the tangent spaces of their respective Lie groups at the identity, and that the mapping of elements

Table 2.1: Lie groups and their attributes.

Group	No. dimensions	Definition
SO(3)	3	$\mathrm{SO}(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times3} \vert \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = 1 \right\}$
SE(3)	6	$\mathrm{SE}(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \vert \mathbf{R} \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\}$
Sim(3)	7	$\mathrm{Sim}(3) = \left\{ \mathbf{S} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \vert \mathbf{R} \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3, s \in \mathbb{R}^+ \right\}$

Table 2.2: Lie groups and their corresponding Lie algebras.

Lie group	Lie algebra	Definition
SO(3)	$\mathfrak{so}(3)$	$\mathfrak{so}(3) = \left\{ \boldsymbol{\phi} \in \mathbb{R}^3 \vert \boldsymbol{\phi}^\wedge = \boldsymbol{\Phi} = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times3} \right\}$
SE(3)	$\mathfrak{se}(3)$	$\mathfrak{se}(3) = \left\{ \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix} \in \mathbb{R}^6 \vert \boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}_{1\times3} & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \vert \boldsymbol{\rho} \in \mathbb{R}^3, \boldsymbol{\phi} \in \mathfrak{so}(3) \right\}$
Sim(3)	$\mathfrak{sim}(3)$	$\mathfrak{sim}(3) = \left\{ \boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \\ \sigma \end{bmatrix} \in \mathbb{R}^7 \vert \boldsymbol{\zeta}^\wedge = \begin{bmatrix} \boldsymbol{\rho}\mathbf{I} + \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}_{1\times3} & 0 \end{bmatrix} \vert \boldsymbol{\rho} \in \mathbb{R}^3, \boldsymbol{\phi} \in \mathfrak{so}(3), \sigma \in \mathbb{R}^+ \right\}$

Note: $(.)^\wedge$ is the hat operator, not to be confused with the homogeneous coordinates representation $(\hat{.})$.

from a Lie group G to the corresponding Lie algebra $\mathfrak{g}$ is called the logarithm map ($\log : \mathrm{G} \to \mathfrak{g}$) and the reversed mapping is called the exponential map ($\exp : \mathfrak{g} \to \mathrm{G}$). For a more in-depth treatment on the topic of Lie groups and Lie algebras, we refer to references [96]–[98].

Jacobians

Now that we have the minimal representation of a transformation matrix (the Lie algebras), we can perform the derivatives of the camera projection function with respect to the 6 degrees of freedom (DOF) of the camera pose. For the camera projection, we define a series of mappings from a world map point to a re-projected pixel in a local frame. Let $\mathbf{p}_w$ be a map point in the world coordinate frame, $\mathbf{T}_{1\to w}$ be the transformation matrix from frame 1 to the world coordinate frame, and $\mathbf{a}_1$ be the projected image coordinates in frame 1. First, we transform $\mathbf{p}_w$ to frame 1:

$$\mathbf{p}'_1 = \underbrace{(\mathbf{T}_{1\to w}\hat{\mathbf{p}}_w)_{1:3}}_{\text{drop last row}} \tag{2.12}$$

Second, we project the local map point $\mathbf{a}'_1$ to the image coordinates:

$$\mathbf{a}_1 = \pi(\mathbf{p}'_1). \tag{2.13}$$

Lastly, we can obtain the image intensity at the projected (sub)pixel image coordinates $I_1(\mathbf{a}_1)$. Once the camera projection is defined, we can calculate the Jacobians of the projection function with respect to the 6 DOF Lie algebra elements $\boldsymbol{\xi}$ using chain rule:

$$\mathbf{J}_{\text{proj}} = \mathbf{J}_I \mathbf{J}_\pi \mathbf{J}_{\mathbf{p}'}, \tag{2.14}$$

where

$$\mathbf{J}_I = \begin{bmatrix} \frac{\partial I}{\partial u} & \frac{\partial I}{\partial v} \end{bmatrix}, \tag{2.15}$$

$$\begin{aligned} \mathbf{J}_\pi &= \begin{bmatrix} \frac{\partial u}{\partial x'} & \frac{\partial u}{\partial y'} & \frac{\partial u}{\partial z'} \\ \frac{\partial v}{\partial x'} & \frac{\partial v}{\partial y'} & \frac{\partial v}{\partial z'} \end{bmatrix} \\ &= \begin{bmatrix} \frac{f_x}{z'} & 0 & -\frac{f_x x'}{z'^2} \\ 0 & \frac{f_y}{z'} & -\frac{f_y y'}{z'^2}, \end{bmatrix} \end{aligned} \tag{2.16}$$

and

$$\begin{aligned} \mathbf{J}_{\mathbf{p}'} &= \left. \frac{\partial \mathbf{p}'}{\partial \boldsymbol{\xi}} \right|_{\boldsymbol{\xi}=0} \\ &= [\mathbf{G}_{tx}\hat{\mathbf{p}}' | \mathbf{G}_{ty}\hat{\mathbf{p}}' | \mathbf{G}_{tz}\hat{\mathbf{p}}' | \mathbf{G}_{rx}\hat{\mathbf{p}}' | \mathbf{G}_{ry}\hat{\mathbf{p}}' | \mathbf{G}_{rz}\hat{\mathbf{p}}'] \\ &= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & z' & -y' \\ 0 & 1 & 0 & -z' & 0 & x' \\ 0 & 0 & 1 & y' & -x' & 0 \end{bmatrix}}_{\text{dropped last row}}. \end{aligned} \tag{2.17}$$

For brevity, we omit the numbering subscripts. The generator matrices $\mathbf{G}_{tx}$, $\mathbf{G}_{ty}$, $\mathbf{G}_{tz}$, $\mathbf{G}_{rx}$, $\mathbf{G}_{ry}$, $\mathbf{G}_{rz}$ and $\mathbf{G}_s$ evaluates the derivatives of the Lie algebra

elements around the identity:

$$\mathbf{G}_{tx} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_{ty} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_{tz} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{G}_{rx} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_{ry} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_{rz} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{G}_{s} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Above is the illustration of Jacobians on the SE(3) manifold, and the derivations can be generalized to SO(3) and Sim(3). In the context of SLAM, calculating the Jacobians of camera projection function has been used in structure and motion [99], [100] as well as propagation of errors (or uncertainty) [37], [101].

2.4 Learning to predict depth from images

This section details the relevant methods for learning single-image depth estimation in this thesis. The methods pertinent to solving our problems are unsupervised (Section 2.4.1) and semi-supervised learning (Section 2.4.2) used for training a CNN.

2.4.1 Unsupervised learning

Unsupervised learning (sometimes also referred to as self-supervised learning) is a method that does not require ground truth depth in training. Rather, unsupervised learning of depth uses *novel view synthesis* to optimize the predicted depth maps [68]–[70]. The key idea is that, with the predicted depth map and known camera transformation between two views, the pixels can be densely projected from one view to another, and the predicted depth dictates the accuracy of the projection to synthesize the novel view.

Unsupervised learning can be performed by minimizing the difference between the target image and its synthesized target images from a set of source images (that are close to the target image). To synthesize a novel view, we reconstruct the target image $I_{j\to i}$ using a source image I_j, the relative transformation $\mathbf{T}_{j\to i}$ and the predicted depth $D_{i,\text{CNN}}$:

$$\begin{aligned} I_{j\to i}(\mathbf{a}) &= I_j(\mathbf{a}') \quad \forall \mathbf{a} \in \Omega_j, \\ \mathbf{a}' &= \pi\left(\mathbf{T}_{j\to i} D_{i,\text{CNN}}(\mathbf{a}) \pi^{-1}(\mathbf{a})\right), \end{aligned} \tag{2.18}$$

Ω_j the set of pixel locations in I_i with valid re-projections in I_j. Then, the per-pixel photometric error can be computed as follows:

$$pe(I_{j\to i}, I_i, \mathbf{a}) = \left\| I_{j\to i}(\mathbf{a}) - I_i(\mathbf{a}) \right\|_1. \tag{2.19}$$

where *pe* is the photometric error function. Following [65], the structural similarity index (SSIM) [102] can be added to the photometric error for improved training accuracy:

$$pe(I_{j\to i}, I_i, \mathbf{a}) = \frac{\alpha}{2}(1 - \text{SSIM}(I_{j\to i}, I_i)(\mathbf{a}) + (1-\alpha) \left\| I_{j\to i}(\mathbf{a}) - I_i(\mathbf{a}) \right\|_1, \tag{2.20}$$

where α controls the relative contribution between the SSIM and absolute intensity difference. Instead of aggregating the photometric errors, Godard et al. [103] propose the use of per-pixel minimum photometric error across the reconstructed target images to minimize the occlusion artefacts around the foreground borders:

$$\mathcal{L}_{\text{photo}} = \frac{1}{|\Omega|} \sum_{\mathbf{a}\in\Omega} \min_j pe(I_i, I_{j\to i}, \mathbf{a}), \tag{2.21}$$

where $j \in \{i-1, i+1\}$ is the image index of the source images, Ω the set of pixel locations in I_i, $|\Omega|$ the total number of pixels and $\mathcal{L}_{\text{photo}}$ the photometric loss term in the training loss. To further improve the training, a *smoothness* loss term $\mathcal{L}_{\text{smooth}}$ is also included in the training loss, which penalizes depth discontinuities in the texture-poor image regions:

$$\mathcal{L}_{\text{smooth}} = \sum_{\mathbf{a}\in\Omega_i} \left(|\partial_x D_{i,\text{CNN}}(\mathbf{a})| e^{-|\partial_x I_i(\mathbf{a})|} + |\partial_y D_{i,\text{CNN}}(\mathbf{a})| e^{-|\partial_y I_i(\mathbf{a})|} \right), \tag{2.22}$$

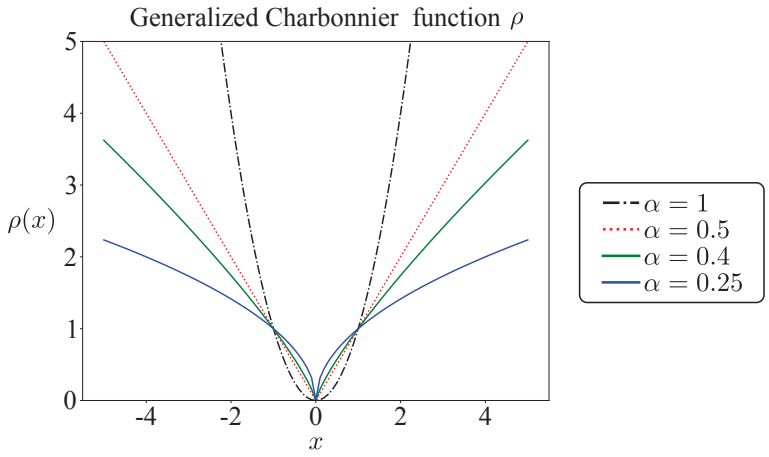

Figure 2.5: Generalized Charbonnier function with different parameter settings.

where ∂ is the gradient operator. Therefore, by combining the photometric loss with the smoothness loss, the unsupervised loss function is given by

$$\mathcal{L}_{\text{unsupervised}} = \mathcal{L}_{\text{photo}} + \lambda \mathcal{L}_{\text{smooth}}, \tag{2.23}$$

where λ is a weighting parameter.

2.4.2 Semi-supervised learning

Semi-supervised learning extends the unsupervised learning with ground truth sparse/dense depth supervision. To this end, we can introduce an additional supervised depth loss term $\mathcal{L}_{\text{depth}}$ that incorporates the depth errors between the predicted depth and the ground truth depth [64], [73]:

$$\mathcal{L}_{\text{depth}} = \sum_{\mathbf{a} \in \Omega_{\text{gt}}} \left\| D_{\text{gt}}^{-1}(\mathbf{a}) - D_{i,\text{CNN}}^{-1}(\mathbf{a}) \right\|_1, \tag{2.24}$$

where Ω_{gt} is a set pixels with valid ground truth depth. Combining the supervised depth loss term $\mathcal{L}_{\text{depth}}$ with the unsupervised loss function (Eq. 2.23), the semi-supervised loss function is given by

$$\mathcal{L}_{\text{semi\_supervised}} = \mathcal{L}_{\text{photo}} + \lambda_1 \mathcal{L}_{\text{smooth}} + \lambda_2 \mathcal{L}_{\text{depth}}, \tag{2.25}$$

where λ_1 and λ_2 are the weighting parameters.

For less reliable ground truth depth (e.g., sparse depth generated by SLAM), we may add a generalized Charbonnier function[3] [104] to alleviate the influence of outliers, which is given by

$$\rho(x) = (x^2 + \epsilon^2)^\alpha, \tag{2.26}$$

where ϵ is a small constant. Setting $\alpha = 0.5$ results in the Charbonner function (a differentiable L1 norm) and that reducing α below 0.5 increases the non-convexity of the function (see Figure 2.5). Therefore, we can rewrite Eq. 2.24 to use the generalized Charbonnier function for better robustness against outliers:

$$\mathcal{L}_{\text{depth}} = \sum_{\mathbf{a} \in \Omega_{\text{gt}}} \left\| D_{\text{gt}}^{-1}(\mathbf{a}) - D_{i,\text{CNN}}^{-1}(\mathbf{a}) \right\|_\rho. \tag{2.27}$$

[3]This is similar to M-estimators (e.g., Huber norm and Tukey norm) used for solving a system of linear equations.

Chapter 3

CNN-SVO: Improving the Mapping in Semi-Direct Visual Odometry Using Single-Image Depth Prediction

3.1 Overview

Reliable feature correspondence between frames is critical in visual SLAM algorithms. In comparison with existing visual SLAM algorithms, semi-direct visual odometry (SVO) adopts *direct* feature correspondence and efficient implementation of probabilistic mapping method, which lead to state-of-the-art frame rate camera motion estimation. However, one main limitation in SVO is the large depth uncertainty when initializing new map points, resulting in increased likelihood of erroneous feature matching (see Figure 3.1) and slow convergence in estimating new map points. This chapter presents a method that improves the SVO mapping by initializing the mean and variance of the depth at a feature location according to the predicted depth from a single-image depth prediction CNN.

The key contribution of this study is to analyze the effectiveness of using single-depth prediction as prior knowledge to improve mapping performance. Formally, given I_i as the current keyframe, I_j as a nearby frame and F as the features initialized in I_i, the goal is to find the correspondences of the features F between I_i and I_j. For each feature location $\mathbf{f}_m \in F$ in I_i, the corresponding

feature $\mathbf{f}'_m$ can be identified along the epipolar line L_m in I_j, which can be defined as follow:

$$L_m := \{\mathbf{p}_{\min} + \alpha \begin{pmatrix} l_{m,x} \\ l_{m,y} \end{pmatrix} \mid \alpha \in S_m\}, \tag{3.1}$$

where $\mathbf{p}_{\min}$ is the image coordinates in I_j that correspond to the minimum depth of the search range, $\begin{pmatrix} l_{i,x} & l_{i,ys} \end{pmatrix}^T$ the normalized direction of the epipolar line, and S_m the search interval of $\mathbf{f}_m$. By constraining the correspondence search along L_m, a correspondence is considered found with the lowest zero-mean sum of squared difference $zmssd$ between $\mathbf{f}_m$ and $\mathbf{f}'_m$ calculated by

$$\mathbf{f}'^*_m = \min_{\mathbf{f}'_m} zmssd(I_i(\mathbf{f}_m), I_j(\mathbf{f}'_m)), \tag{3.2}$$

where $\mathbf{f}'^*_m$ is the image coordinates of the corresponding feature, which is used to triangulate a new depth measurement. Thus, the accuracy of the correspondences is strongly dependant on the depth range S_m being used in the epipolar line search, as more extensive search range may introduce the number of false matches (see Figure 1.8). In Section 3.2, we introduce our proposed solution to improve the correspondence search accuracy.

3.2 Method

For a complete understanding of how the depth range affects the overall accuracy of SVO mapping, we cover the fundamental of SVO mapping algorithm in Section 3.2.1. Next, we detail our improved initialization of the map points in Section 3.2.2.

3.2.1 Review of the SVO mapping algorithm

As shown in Figure 3.2, SVO [50] contains two threads running in parallel: tracking thread and mapping thread. In the tracking thread, the camera pose of a new frame is obtained by minimizing the photometric residuals between the reference image patches (from which the map points are back-projected) and the image patches that are centred at the reprojected locations in the new frame. Concurrently, the mapping thread creates new map points using two

processes: initialization of new map points with large depth uncertainty and update of depth uncertainty of the map points with *depth-filters* [50], [105]; consequently, a new map point is inserted in the map if the depth uncertainty of the map point is small.

Given the camera poses of two frames, one of which is a keyframe, the depth of a feature can be obtained using the following two steps: finding the feature correspondence along the epipolar line in the non-keyframe, and then recovering the depth via triangulation. Since the occurrence of outlier matching is inevitable, a *depth-filter* is run for every new feature, which models the distribution of two types of measurements: inlier and outlier [50], [105]. The first dimension describes the probability distribution of the depth, and the second dimension models the outlier measurement. Therefore, given a set of depth measurements, a *depth-filter* approximates the mean depth and the variance of the feature while determining the confidence of the approximation. The depth uncertainty (i.e., approximated variance) of the feature is updated when there is a new depth measurement, and the *depth-filter* is considered to have converged if the updated depth uncertainty is below a threshold. Then, the converged *depth-filters* that contain the true depths are used to create new map points by back-projecting the points at their feature locations according to the converged depth.

However, SVO mapping initializes new map points in a reference keyframe with large uncertainty, and their mean depths are set to the average scene depth in the reference frame. While such an initialization strategy is reasonable for the scene with one dominant plane (e.g., the floor plane), the large depth uncertainty has limited the capability of the mapping to determine the true depths of the map points for the scene in which the depths of the map points vary considerably. Mainly, large depth uncertainty introduces two problems: (1) possible erroneous feature correspondence along the epipolar line in the nearby frames and (2) a high number of depth measurements required to converge to the true depth.

Given a set of triangulated depth measurements, the goal of using *depth-filter* is to separate the good measurements from the bad measurements; inlier

measurements are normally distributed around the true depth, whereas outlier measurements are uniformly distributed within an interval $[\rho_i^{\min}, \rho_i^{\max}]$. Specifically given a set of triangulated inverse depth measurements $\{\rho_i^1, \rho_i^2, \ldots, \rho_i^N\}$ that correspond to the same feature, the measurement ρ_i^n is modeled in SVO using a *Gaussian + Uniform* mixture model:

$$p(\rho_i^n|\rho_i, \gamma_i) = \gamma_i \mathcal{N}(\rho_i^n|\rho_i, \tau_i^2) + (1 - \gamma_i)\mathcal{U}(\rho_i^n|\rho_i^{\min}, \rho_i^{\max}), \tag{3.3}$$

where ρ_i is the true inverse depth, τ_i^2 the variance of the inverse depth, and γ_i the inlier ratio. Assuming the inverse depth measurements are independent, the approximation of the true inverse depth posterior can be computed incrementally by the product of a Gaussian distribution for the depth and a Beta distribution for the inlier ratio [50]:

$$q(\rho_i, \gamma_i|a_n, b_n, \mu_n, \sigma_n^2) = Beta(\gamma_i|a_n, b_n)\mathcal{N}(\rho_i|\mu_n, \sigma_n^2) \tag{3.4}$$

where a_n and b_n are the parameters in the Beta distribution, and μ_n and σ_n^2 the mean and variance of the Gaussian depth estimate. The incremental Bayesian update step for a_n, b_n, μ_n, and σ_n^2 is described in detail in [50], [105]. Once σ_n^2 is lower than a threshold, the *depth-filter* has converged to the true depth. The converged true depth is then used to create a new map point.

3.2.2 Proposed method

To reduce the depth uncertainty, we propose to initialize new *depth-filters* with depth prior from a single-image depth prediction CNN [65] (i.e., small variance centred about the predicted depth), such that the likelihood of identifying the corresponding features is vastly increased (see Figure 3.1). Combining existing SLAM algorithms with single-image depth estimation from a CNN has been proposed to overcome the limitation of depth uncertainty. Notably, such combinations have been designed to tackle two classes of problems: dense reconstruction [14], [15] and map scale recovery [12], [106]. One advantage of these combinations is to initialize the depth search within the optimal convergence basin using depth information from a CNN.

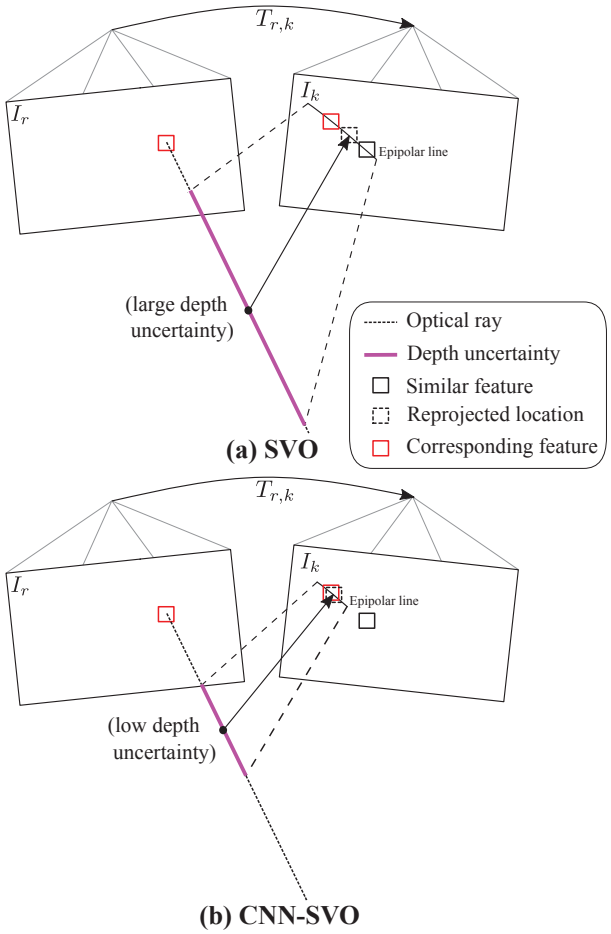

Figure 3.1: Proposed *depth-filter* initialization strategy. Each initialized *depth-filter* has a mean depth (black dot) and an interval in which the corresponding feature should lie, as shown by the magenta line. Note that larger depth uncertainty can allow the erroneous match to happen (as illustrated in (a) where the depth filter could converge to the "similar feature" rather than the "corresponding feature"). Our proposed *depth-filter* initialization method using depth estimation from a convolutional neural network (CNN) (see (b)) has lower depth uncertainty for identifying the corresponding feature.

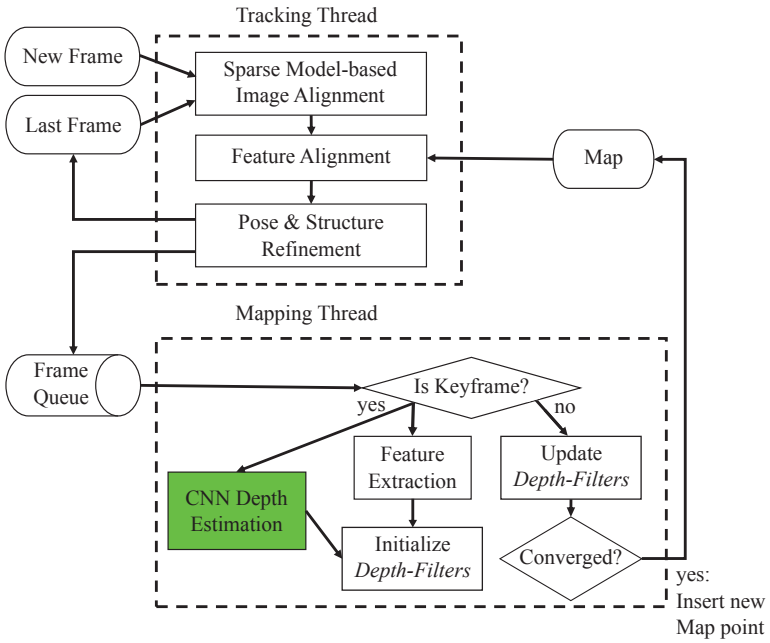

Figure 3.2: The CNN-SVO pipeline. Our work augments the SVO pipeline [50] with the CNN depth estimation module (marked in green) to improve the mapping in SVO.

With single-image depth prediction as prior knowledge of the scene geometry, our CNN-SVO can obtain a better estimate of the mean and a smaller initial variance of a *depth-filter* than SVO to allow for the convergence of a map point to the true depth. Figure 3.2 illustrates the CNN-SVO pipeline, in which we add the CNN depth estimation module (marked in green) to provide strong depth priors in the map points initialization process when a keyframe is selected—the initialization of *depth-filters*.

Hence, each *depth-filter* is initialized with the following parameters: the mean of the inverse depth μ_n and the variance of the inverse depth σ_n^2. Table 3.1 compares the initialization of the parameters between SVO and CNN-SVO. The key difference is that CNN-SVO initializes the feature's mean and variance using learned scene depth instead of using the average and minimum scene depths in the reference keyframe. Setting the proper variance is again a crucial design step in the experiment, as a small variance provides adequate room for noisy depth prediction to converge. We empirically found that setting the depth variance to $\frac{1}{(6d_{\text{CNN}})^2}$ provides adequate room for noisy depth predic-

Table 3.1: A comparison between SVO and CNN-SVO in the initialization of parameters. The parameters are defined by prior knowledge of the scene, where d_{avg} is the average scene depth in the reference keyframe, d_{CNN} the depth prediction from the single-image depth prediction CNN, d_{min} the minimum scene depth in the reference keyframe, μ_n the mean of the feature's inverse depth, and σ_n^2 the variance of the feature's inverse depth.

	SVO	CNN-SVO
μ_n	$\frac{1}{d_{\text{avg}}}$	$\frac{1}{d_{\text{CNN}}}$
σ_n^2	$\frac{1}{(6d_{\text{min}})^2}$	$\frac{1}{(6d_{\text{CNN}})^2}$

tion to converge; we will be losing the absolute scale if the depth variance is large (e.g., replacing 6 with a higher number) by allowing more uncertainty in the measurement. Based on the initialized μ_n and σ_n^2, a depth interval $[\rho_i^{\text{min}}, \rho_i^{\text{max}}]$ can be defined by

$$\rho_i^{\text{min}} = \mu_n + \sqrt{\sigma_n^2}, \tag{3.5}$$

$$\rho_i^{\text{max}} = \begin{cases} 0.00000001, & \text{if } \mu_n - \sqrt{\sigma_n^2} < 0 \quad (3.6a) \\ \mu_n - \sqrt{\sigma_n^2}, & \text{otherwise.} \quad (3.6b) \end{cases}$$

so that the corresponding feature can be found in the limited search range along the epipolar line in the nearby views (see Figure 3.1). By obtaining strong depth prior from the single-image depth prediction CNN, the benefits are twofold: smaller uncertainty in identifying feature correspondence and faster map point convergence, as illustrated in Figure 3.3.

3.3 Implementation

To provide depth prediction in the initialization of map points in CNN-SVO, we adopt the Resnet50 variant of the encoder-decoder architecture from [65] that has already been trained on Cityscape dataset. Next, we fine-tune the network on stereo images in KITTI raw data excluding KITTI Odometry Sequence 00-10 using original settings in [65] for 50 epochs. To produce consistent structural information, even on overexposed or underexposed images, the brightness of the images has been randomly adjusted throughout the training,

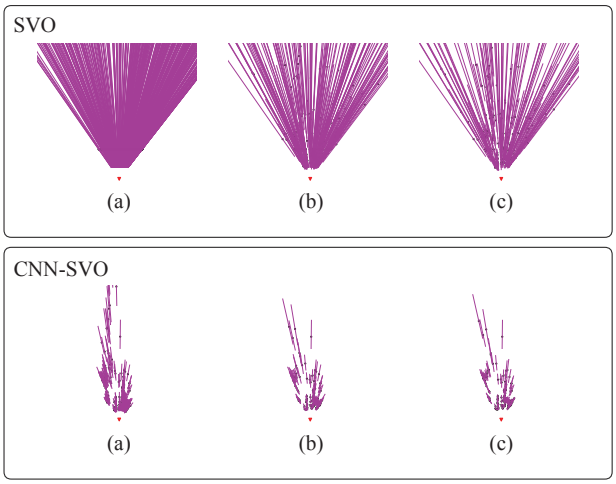

Figure 3.3: Preliminary results of the *depth-filters* from images with ground truth camera poses obtained from KITTI dataset [16]. The length of the magenta line represents the depth uncertainty and the triangle at the bottom is the centre of projection. (a) initialization of the *depth-filters* where SVO uses a large interval to model the uncertainty of each initial map point whereas CNN-SVO uses a short interval; (b) depth estimates of the map points by the *depth-filters* after three updates; (c) depth estimates of the map points by the *depth-filters* after five updates.

creating the effect of illumination variation. This consideration is helpful for the CNN to handle high dynamic range (HDR) environments (see Fig. 3.4).

To design the system with real-time capability, we resize the images to 512×256 for depth map inference and back to the original shape for SLAM processing. While two separate threads have been designed to handle mapping and tracking, GPU is used to provide the depth maps for the keyframes. The hardware is an Intel i7 processor[1] with an NVidia GeForce GTX Titan X graphics card.

To scale the depth prediction for other datasets, the scaled depth d_{current} can be obtained by the inferred depth d_{trained} multiplied by the ratio of current focal length f_{current} to trained focal length f_{trained}, i.e.,

$$d_{\text{current}} = \frac{f_{\text{current}}}{f_{\text{trained}}} d_{\text{trained}}. \tag{3.7}$$

We set the maximum and the minimum number of tracked features in a

[1]Intel i7-4790K, 4 cores, 4.0GHz, 32GB RAM

frame to 200 and 100, respectively. Regarding the *depth-filter*, we modify SVO to use five previous keyframes to increase the number of measurements in the *depth-filters*. We also enable bundle adjustment during the evaluation process.

3.4 Results

For the evaluation, we calculate the absolute trajectory RMSEs[2] (ATEs) on the KITTI [16] and Oxford Robotcar [17] datasets. The ATEs of KITTI and Oxford Robotcar datasets are collected with a median of five runs to account for the non-deterministic factors of running the visual SLAM algorithms.

We use eleven KITTI Odometry sequences and nine Oxford Robotcar sequences for performance benchmarking. As for the images, we use the left camera from KITTI binocular stereo setup and the centre camera of the Bumblebee XB3 trinocular stereo setup from Oxford Robotcar. Both of the image streams are captured using global shutter cameras. Note that the ground truth poses from Oxford Robotcar dataset are not reliable for evaluation [17], because of the poor and inconsistent GPS signals; we still use the ground truth for both quantitative and qualitative evaluation purposes. The frame rates are 10 frames per second (FPS) and 16 FPS for KITTI and Oxford Robotcar, respectively. To maintain the same aspect ratio that is used by the network input, the images in the Oxford Robotcar dataset have been cropped to 1248x376 throughout the evaluation process. We skip the first 200 frames for all the Oxford Robotcar sequences because of the extremely overexposed images at the beginning of the sequences.

We compare our proposed method against the state-of-the-art SLAM algorithms, namely SVO [50], DSO [38] and ORB-SLAM without loop closure [33]. In addition, we indicate with 'X' for methods that are unable to complete the sequence due to lost tracking in the middle of the sequence. In Section 3.4.1, we compare the camera tracking accuracy between CNN-SVO and the state-of-the-art visual SLAM algorithms. In Section 3.4.2. we evaluate the runtime performance of CNN-SVO. In Section 3.4.3, we evaluate the scale factors of

[2]root mean square error

the evaluated trajectories against the ground truth trajectories.

3.4.1 Accuracy evaluation

Table 3.2 show that CNN-SVO outperforms the state-of-the-art SLAM algorithms on the KITTI dataset. Overall, our system can track all the sequences except for KITTI Sequence 01 because of failure to match features accurately in the scene with repetitive structure. On the other hand, SVO is designed to perform well in a planar scene; therefore, it fails to identify corresponding features effectively in the outdoor scene, where the depths of the features can vary considerably. Note that the large ATEs by DSO and ORB-SLAM are due to scale drift (see Figure 3.5 (a) for a qualitative comparison).

Table 3.3 compares the ATEs on the Oxford Robotcar dataset. We demonstrate that our competitors fail to track most of the Oxford Robotcar sequences containing severely underexposed and overexposed images. Notably, ORB-SLAM failed to match features when the textural information in the images vanishes; the tracking failure in DSO may be due to the inability of affine brightness modelling to handle severe brightness change in the sequences, a problem that has also been reported in stereo DSO [107]. We attribute the robust tracking of CNN-SVO to its ability to match features in consecutive frames with additional depth information, even when the images are overexposed or underexposed (see Figure 3.4). The experimental results suggest generalizability to a structurally similar scene since the CNN has not been trained on Oxford Robotcar sequences.

The qualitative comparison of the camera trajectories can be found in Figure 3.5 for KITTI dataset and Robotcar dataset, respectively. In Figure 3.5 (b), an S-like curve is produced by CNN-SVO near the end of trajectory in Sequence 2014-05-06-12-54-54, which is caused by a moving car in front of the camera.

3.4.2 Runtime evaluation

Local BA (about 29 ms) and single-image depth prediction (about 37 ms) have been the most demanding processes in the pipeline, but both processes

Table 3.2: Absolute keyframe trajectory error (in metre) on the KITTI dataset [16].

Sequence	SVO	CNN-SVO	DSO	ORB-SLAM (w/o loop closure)
00	X	**17.5269**	113.1838	77.9502
01	X	X	X	X
02	X	50.5119	116.8108	**41.0064**
03	X	3.4588	1.3943	**1.0182**
04	58.3970	2.4414	**0.422**	0.9302
05	X	**8.1513**	47.4605	40.3542
06	X	**11.5091**	55.6173	52.2282
07	X	**6.5141**	16.7192	16.546
08	X	**10.9755**	111.0832	51.6215
09	X	**10.6873**	52.2251	58.1742
10	X	**4.8354**	11.090	18.4765

Table 3.3: Absolute keyframe trajectory error (in metre) on the Oxford Robotcar dataset [17].

Sequence	SVO	CNN-SVO	DSO	ORB-SLAM (w/o loop closure)
2014-05-06-12-54-54	X	8.657	**4.708**	10.6596
2014-05-06-13-09-52	X	**9.1947**	X	X
2014-05-06-13-14-58	X	**10.1865**	X	X
2014-05-06-13-17-51	X	**8.26**	X	X
2014-05-14-13-46-12	X	**13.7513**	X	X
2014-05-14-13-50-20	X	**32.4199**	X	X
2014-05-14-13-53-47	X	**6.3017**	X	X
2014-05-14-13-59-05	X	6.1515	**2.4532**	X
2014-06-25-16-22-15	X	**3.703**	X	6.558

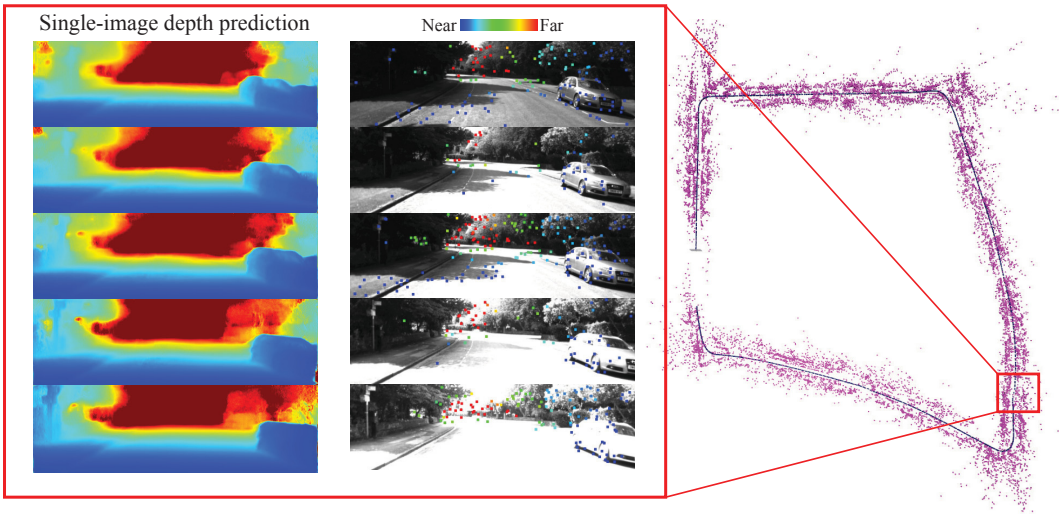

Figure 3.4: CNN-SVO: Camera motion estimation in the high dynamic range (HDR) environment. Left: The single-image depth prediction CNN demonstrates the illumination invariance property in estimating depth maps, and the colour-coded reprojected map points on the five consecutive frames show the reprojected map points onto those frames (best viewed in colour). Right: Camera trajectory and map points in magenta generated by CNN-SVO.

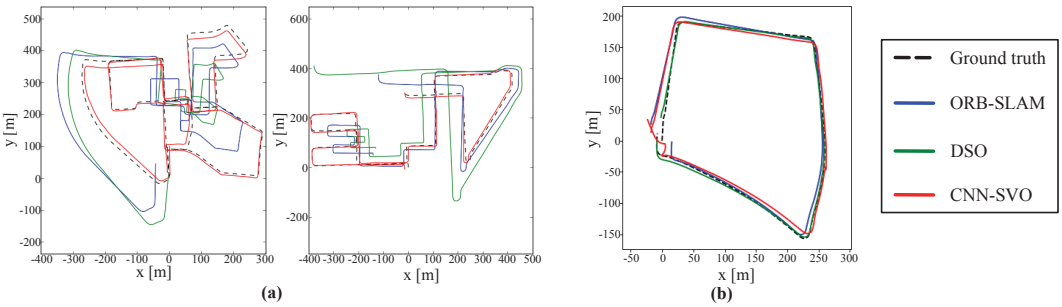

Figure 3.5: Qualitative comparison of camera trajectories produced by ORB-SLAM (without loop closure), DSO, and CNN-SVO. SVO is not included in this figure because it is not able to complete the trajectory due to tracking and mapping failures. (a) KITTI odometry sequence 00 and 08; (b) Oxford Robotcar Sequence 2014-05-06-12-54-54.

are only required when new keyframes are created. Despite the computational demand, we experimentally found that CNN-SVO runs faster at 16 FPS with Oxford Robotcar dataset than 10 FPS with the KITTI dataset due to the close distance between frames in high frame rate; hence lesser keyframes are selected relative to the total number of frames from the sequence. For this reason, the real-time computation can be achieved.

3.4.3 Scale evaluation

Since the network is trained on rectified stereo images with a known static stereo baseline, we examine the scale of the odometry based on predicted depth from the network. From Table 3.4, the average recovered scale is 1.0183 and 0.9313 with a standard deviation of 0.0793 and 0.0231, for the KITTI and Oxford Robotcar datasets, respectively. As expected, CNN-SVO is able to achieve close to metric scale in the evaluated sequences, as the training data is based on the scene on the KITTI dataset. As for the Oxford Robotcar dataset, we offer two possible explanations for the inconsistent odometry scale. First, as mentioned in the Oxford Robotcar dataset documentation, the provided ground truth poses are not accurate because of the inconsistent GPS signals and scale drift in the large-scale map (see Section III in [17]). Second, the single-image depth prediction network has not been trained on the Oxford Robotcar dataset images, so the absolute scale recovery cannot be guaranteed.

3.5 Summary

The key takeaways of this chapter are as follows.

- We propose a method that improves the mapping performance of the state-of-the-art semi-direct visual odometry, which we refer to as CNN-SVO.
- Using single-image depth prediction as a prior, features can be matched effectively by limiting the search range along the epipolar line in nearby views, assuming the camera poses are known.

Table 3.4: Scale factor of the evaluated trajectories on the (left) KITTI dataset [16] and (right) Oxford Robotcar Dataset [17].

Sequence	Scale
Sequence 00	0.9296
Sequence 01	X
Sequence 02	0.921
Sequence 03	1.0811
Sequence 04	1.1876
Sequence 05	0.9837
Sequence 06	0.9602
Sequence 07	1.0246
Sequence 08	1.0014
Sequence 09	1.043
Sequence 10	1.0512

Sequence	Scale
2014-05-06-12-54-54	0.8953
2014-05-06-13-09-52	0.9321
2014-05-06-13-14-58	0.9172
2014-05-06-13-17-51	0.9399
2014-05-14-13-46-12	0.9103
2014-05-14-13-50-20	0.9737
2014-05-14-13-53-47	0.9427
2014-05-14-13-59-05	0.9473
2014-06-25-16-22-15	0.9236

- Also, *depth-filters* are initialized with low depth uncertainty, and therefore they can converge to their true depth values more effectively than the original SVO formulation in order to create new map points.

- With the improved mapping performance, experimental results indicate that CNN-SVO has better camera tracking accuracy than the state-of-the-art monocular SLAM algorithms.

Chapter 4

DeepRelativeFusion: Dense Monocular SLAM using Single-Image Relative Depth Prediction

4.1 Overview

In Chapter 3, we describe the idea of using single-image depth prediction as a strong prior in identifying feature correspondences. Despite improved feature matching, the reconstructed map in CNN-SVO is sparse. Similar to the other state-of-the-art SLAM algorithms reviewed in Chapter 1, they all produce a sparse or semi-dense map. This chapter extends the state-of-the-art direct SLAM algorithm by densifying the semi-dense map to generate a dense map.

4.2 Related work

Traditional monocular SLAM algorithms are capable of producing sparse, semi-dense, and dense structures. Conceptually, sparse refers to the sparsity of the structure as well as the independence of each space point from one another during the structure and motion optimization. During the optimization, each image point (usually a corner) is being matched across frames and mapped, and collectively, the whole structure and camera motion are being optimized through photometric [38] or geometric [33], [50] re-projection error minimization. On the other hand, instead of processing the sparse points inde-

pendently, semi-dense and dense methods employ the notion of the neighbourhood *connectedness* of the points. Dense methods regularize the neighbouring depth pixels using image gradient [49], [108], [109], typically formulated as a *smoothness* term in an energy minimization framework; whereas the semi-dense method, LSD-SLAM [37], estimates the depth values of the high gradient image regions, thus semi-dense, and regularizes the semi-dense depth map by computing a weighted average of in the neighbouring depth values with the estimated variances as their weight. In this work, we use LSD-SLAM to recover a semi-dense structure reliably. Next, we filter the semi-dense structure using contextual information of the local photometric and depth information, which is inspired by the edge-preserving bilateral filtering from Tomasi and Manduchi [110]. Then, we perform densification by regularizing the structure using the filtered semi-dense structure, and depth and depth gradient information from single-image depth prediction.

Fusions of single-image depth prediction to visual SLAM algorithms have been proposed to solve dense reconstruction problems. One approach to performing depth fusion from multiple viewpoints is through the accumulation of probabilistic distribution of depth observations from the single-image depth prediction [15], [111]. Recently, Czarnowski et al. propose a factor-graph optimization framework named DeepFactors [112], which jointly optimizes the camera motion and the code-based depth maps. Each depth map is parameterized in an n-dimensional code to avoid costly per-pixel depth map optimization. Another dense SLAM system proposed by Laidlow et al., named DeepFusion [89], uses the depth and depth gradient predictions from a CNN to constrain the optimized depth maps. Our proposed system is similar to DeepFusion, except for three key differences:

1. We use depth and depth gradient from relative depth prediction as priors in the densification of semi-dense depth maps generated by LSD-SLAM.

2. Through extensive experimentation, we have a better cost function for performing densification than DeepFusion.

3. We use the densified depth maps to refine the camera pose.

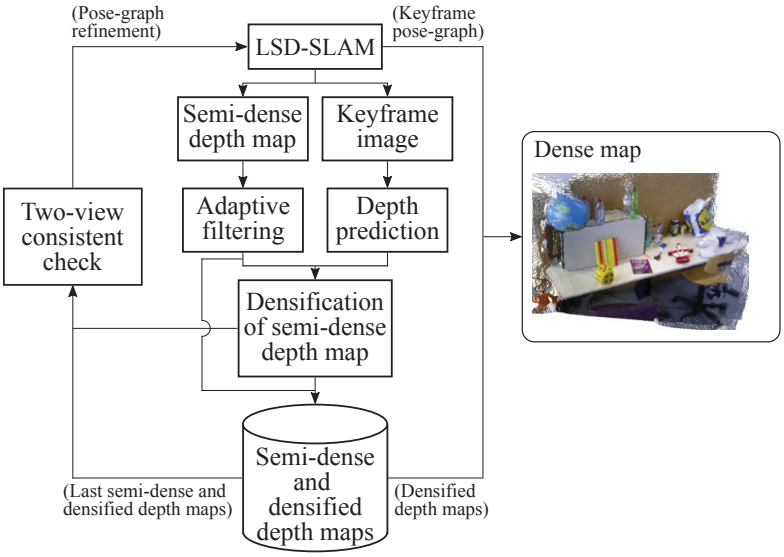

Figure 4.1: Our dense monocular SLAM sytem. We introduce a depth prediction module, an adaptive filtering module and a dense mapping module to the state-of-the-art semi-dense SLAM pipeline, LSD-SLAM [37]. The optimized depth maps are being used to improve pose-graph optimization, while the optimized pose-graph combines with the densified depth maps to generate a globally consistent 3D reconstruction.

4.3 Method

Figure 4.1 illustrates our proposed dense SLAM algorithm. The proposed algorithm contains an optimization framework using the predicted keyframe depth maps (see Section 4.3.1) and the filtered semi-dense depth maps (see Section 4.3.2) to perform densification (see Section 4.3.3). The optimized depth maps are, in turn, used to optimize the keyframe pose-graph in LSD-SLAM (see Section 4.3.4). For dense scene reconstruction, we back-project the densified depth maps from their respective keyframe poses from the optimized keyframe pose-graph.

4.3.1 Depth prediction

For every new keyframe $\mathcal{K}_i$, we predict a relative depth map using MiDaS[1] [82] to densify the semi-dense depth map. Because $D_{i,\text{CNN}}$ is a relative depth

[1]Note that MiDaS v2.0 was the state-of-the-art and was being used to run experiments. There are more accurate pre-trained models available on their official website (`https://github.com/intel-isl/MiDaS`) at the time of preparing the thesis.

map, it needs to be scale- and shift-corrected before it can be used in the densification step. The scale- and shift-correction can be performed as follows:

$$D'_{i,\text{CNN}} = aD_{i,\text{CNN}} + b, \tag{4.1}$$

where $a \in \mathbb{R}^+$ and $b \in \mathbb{R}$ are the scale and shift parameters, respectively. Let $\vec{\mathbf{d}}_n = \begin{pmatrix} d_n & 1 \end{pmatrix}^T$ and $\mathbf{h}^{\text{opt}} = \begin{pmatrix} a & b \end{pmatrix}^T$, and the parameters a and b can be solved in closed-form as follows [82]:

$$\mathbf{h}^{\text{opt}} = \left(\sum_{n \in \Omega_i} \vec{\mathbf{d}}_n \vec{\mathbf{d}}_n^{\,T} \right)^{-1} \left(\sum_{n \in \Omega_i} \vec{\mathbf{d}}_n d'_n \right), \tag{4.2}$$

where $d_n \in D_{i,\text{semi-dense}}$ and $d'_n \in D_{i,\text{CNN}}$ are the inverse depth values of the semi-dense depth map and relative depth map, respectively. Since we will perform scale-and shift-correction for all predicted depth maps, we drop the prime superscript on $D_{i,\text{CNN}}$ for brevity.

4.3.2 Adaptive filter on a semi-dense structure

Our proposed adaptive filter is based upon the bilateral filter [110], which combines the local pixel values according to the geometric closeness $w_d(\cdot,\cdot)$ and the intensity similarity $w_s(\cdot,\cdot)$ between the centre pixel $\mathbf{x}$ and a nearby pixel $\mathbf{x}_n$ within a window $\mathcal{N}$ of an image:

$$\begin{aligned} I_{\text{filtered}}(\mathbf{x}) = \frac{1}{W_{\mathcal{N}}} \sum_{n \in \mathcal{N}} \Bigg(I(\mathbf{x}_n) \\ \underbrace{\exp\Big(-\frac{(I(\mathbf{x}) - I(\mathbf{x}_n))^2}{2\sigma_s^2} \Big)}_{=: w_s(\mathbf{x}, \mathbf{x}_n)} \underbrace{\exp\Big(-\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{2\sigma_d^2} \Big)}_{=: w_d(\mathbf{x}, \mathbf{x}_n)} \Bigg) \end{aligned} \tag{4.3}$$

with

$$W_{\mathcal{N}} = \sum_{n \in \mathcal{N}} w_s(\mathbf{x}, \mathbf{x}_n) w_d(\mathbf{x}, \mathbf{x}_n). \tag{4.4}$$

To perform semi-dense depth map filtering, we introduce two weighting schemes, depth uncertainty $w_u(\cdot)$ and CNN depth consistency $w_c(\cdot,\cdot)$, to remove the semi-dense depth pixels that have a large depth uncertainty and

large local variance relative to their corresponding CNN depth:

$$\begin{aligned} w_u(\mathbf{x}_n) &= \exp\left(-\frac{\sigma_u V_{i,\text{semi-dense}}(\mathbf{x}_n)}{D_{i,\text{semi-dense}}(\mathbf{x}_n)^4}\right) \\ w_c(\mathbf{x},\mathbf{x}_n) &= \exp\left(-\frac{\left(\frac{D_{i,\text{semi-dense}}(\mathbf{x})}{D_{i,\text{semi-dense}}(\mathbf{x}_n)} - \frac{D_{i,\text{CNN}}(\mathbf{x})}{D_{i,\text{CNN}}(\mathbf{x}_n)}\right)^2}{2\sigma_c^2}\right), \end{aligned} \tag{4.5}$$

where the *squared ratio difference* in $w_c(\cdot,\cdot)$ computes the scale-invariant error [113], and $D_{i,\text{semi-dense}}(x_n)$ in $w_u(\cdot)$ attenuates spurious depth pixels. σ_s, σ_d, σ_c and σ_u are the tunable weights in their respective spatial kernels. Consequently, we compute the filtered semi-dense depth map $D'_{i,\text{semi-dense}}$ as follows:

$$\begin{aligned} D'_{i,\text{semi-dense}}(\mathbf{x}) = \frac{1}{W'_{\mathcal{N}}} \sum_{n\in\mathcal{N}} \Big(& D_{i,\text{semi-dense}}(\mathbf{x}_n) \\ & w_s(\mathbf{x},\mathbf{x}_n) w_d(\mathbf{x},\mathbf{x}_n) w_c(\mathbf{x},\mathbf{x}_n) w_u(\mathbf{x}_n) \Big) \end{aligned} \tag{4.6}$$

with

$$W'_{\mathcal{N}} = \sum_{n\in\mathcal{N}} w_s(\mathbf{x},\mathbf{x}_n) w_d(\mathbf{x},\mathbf{x}_n) w_c(\mathbf{x},\mathbf{x}_n) w_u(\mathbf{x}_n). \tag{4.7}$$

With the updated $D'_{i,\text{semi-dense}}$, we re-estimate the corresponding semi-dense depth variance map $V'_{i,\text{semi-dense}}$ by taking the weighted average of squared deviations within the local window for all valid semi-dense depth pixels:

$$\begin{aligned} V'_{i,\text{semi-dense}}(\mathbf{x}) = \frac{|\mathcal{N}|}{n_{\text{valid}}} \frac{1}{W'_{\mathcal{N}}} \sum_{n\in\mathcal{N}} \Big(& W'_{\mathcal{N}}(\mathbf{x}_n) \\ & \big(D'_{i,\text{semi-dense}}(\mathbf{x}) - D_{i,\text{semi-dense}}(\mathbf{x}_n)\big)^2 \Big), \end{aligned} \tag{4.8}$$

where $|\mathcal{N}|$ is the total number of pixels within the window, n_{valid} the number of pixels containing depth values, and $W'_{\mathcal{N}}(\cdot)$ the weight computed at a nearby pixel. To ensure the depth variance has a similar weighting effect in densification, we rescale the semi-dense depth variance $V'_{i,\text{semi-dense}}$:

$$V'_{i,\text{semi-dense}} = \frac{\overline{V_{i,\text{semi-dense}}}}{\overline{V'_{i,\text{semi-dense}}}} V'_{i,\text{semi-dense}}, \tag{4.9}$$

where $\bar{\cdot}$ is the mean operator. Then, we remove the noisy depth pixel by excluding the semi-dense depth pixels whose variance is higher than a threshold γ. For ease of notation, we denote the filtered semi-dense depth map and semi-dense variance map as $D_{i,\text{semi-dense}}$ and $V_{i,\text{semi-dense}}$, respectively.

4.3.3 Densification of the semi-dense structure

Let $D_{i,\text{opt}}$ be the densified depth map. We perform densification of $D'_{i,\text{semi-dense}}$ of $\mathcal{K}_i$ using $D'_{i,\text{CNN}}$ through the minimization of the cost function given by:

$$E_{\text{total}} = E_{\text{CNN\_grad}} + \lambda E_{\text{semi-dense}}. \tag{4.10}$$

The first term, $E_{\text{CNN\_grad}}$, is similar to scale-invariant mean squared error used in [113], which enforces depth gradient consistency between $D_{i,\text{CNN}}$ and $D_{i,\text{opt}}$:

$$E_{\text{CNN\_grad}} = \frac{1}{|\Omega|} \sum_{\mathbf{x} \in \Omega} \frac{\left(E_{\text{CNN\_grad},x}(\mathbf{x})\right)^2 + \left(E_{\text{CNN\_grad},y}(\mathbf{x})\right)^2}{\left(1/D_{i,\text{CNN}}(\mathbf{x})\right)^2}, \tag{4.11}$$

with

$$\begin{aligned} E_{\text{CNN\_grad},x} &= \partial_x \ln D_{i,\text{opt}} - \partial_x \ln D'_{i,\text{CNN}} \\ E_{\text{CNN\_grad},y} &= \partial_y \ln D_{i,\text{opt}} - \partial_y \ln D'_{i,\text{CNN}}, \end{aligned} \tag{4.12}$$

where $|\Omega|$ is the cardinality of Ω, and ∂ the gradient operator. The denominator $\left(1/D_{i,\text{CNN}}\right)^2$ in Equation (4.11) simulates the variance of the depth prediction, which provides stronger depth gradient regularization to closer objects than farther objects.

The second term, $E_{\text{semi-dense}}$, minimizes the difference between the optimized depth map and the semi-dense depth map from LSD-SLAM (similar to [89]):

$$E_{\text{semi-dense}} = \frac{1}{|\Omega_i|} \sum_{\mathbf{x} \in \Omega_i} \rho \left(\frac{\left(D_{i,\text{opt}}(\mathbf{x}) - D_{i,\text{semi-dense}}(\mathbf{x})\right)^2}{V_{i,\text{semi-dense}}(\mathbf{x})} \right), \tag{4.13}$$

where $|\Omega_i|$ is the cardinality of Ω_i. We use the generalized Charbonnier penalty function [114], $\rho(.)$, to improve reconstruction accuracy.

4.3.4 Pose-graph refinement

Next, we incorporate the optimized semi-dense and dense depth maps into pose-graph optimization[2]. To minimize the influence of erroneous regions in the structure, we perform a two-view consistency check between the current keyframe $\mathcal{K}_i$ and the last keyframe $\mathcal{K}_{i-1}$ by projecting the semi-dense

[2]Pose-graph optimization is explained in Appendix-B.2

$D_{i-1,\text{semi-dense}}$ and densified $D_{i-1,\text{opt}}$ depth maps from the last keyframe to the current keyframe's viewpoint:

$$\hat{\tilde{\mathbf{x}}} = s\mathbf{K}\mathbf{R}D_{i,\bullet}(\mathbf{x})\pi^{-1}(\hat{\mathbf{x}}) + \mathbf{t} \tag{4.14}$$

with

$$\begin{gathered} \{\mathbf{x}|D_{i-1,\bullet}(\mathbf{x}) > 0\}, \\ \mathbf{S}_{i-1\to i} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \end{gathered} \tag{4.15}$$

and

$$\tilde{D_{i,\bullet}}(\tilde{\mathbf{x}}) = \left[\hat{\tilde{\mathbf{x}}}\right]_3, \tag{4.16}$$

where $\tilde{\mathbf{x}}$ is the warped coordinates with $\hat{\tilde{\mathbf{x}}}$ being its homogeneous coordinates, $\mathbf{S}_{i-1\to i} \in \text{Sim}(3)$ the relative transformation and $\mathbf{K}$ the camera intrinsics. $D_{i-1,\bullet}$ is a placeholder for $D_{i-1,\text{semi-dense}}$ and $D_{i-1,\text{opt}}$, and $\tilde{D_{i,\bullet}}(\tilde{\mathbf{x}})$ the warped $D_{i-1,\text{semi-dense}}$ and $D_{i-1,\text{opt}}$ in $\mathcal{K}_i$'s viewpoint. To retain the semi-dense structure in LSD-SLAM, we exclude the semi-dense depth regions in $D_{i-1,\text{semi-dense}}$ when warping $D_{i-1,\text{opt}}$. The two-view consistent depth map $D_{i,\text{c}}$ is computed as follows:

$$D_{i,\text{c}}(\mathbf{x}) = \begin{cases} D_{i,\bullet}(\mathbf{x}) & \text{if } \left|D_{i,\bullet}(\mathbf{x}) - \hat{D_{i,\bullet}}(\mathbf{x})\right| < \tau_e \\ 0 & \text{otherwise} \end{cases}. \tag{4.17}$$

Consequently, the two-view consistent depth $D_{i,c}$ and depth variance $V_{i,c}$ maps contain a mixture of semi-dense depth ($D_{i-1,\text{semi-dense}}$, $V_{i-1,\text{semi-dense}}$) and densified depth ($D_{i,opt}$, $V_{i,opt}$) regions. Then, we approximate the uncertainty $V_{i,opt}$ associated with $D_{i,\text{opt}}$ by propagating the pose uncertainty $\mathbf{\Sigma}_{\boldsymbol{\xi},i} \in \mathbb{R}^{7\times 7}$ estimated by LSD-SLAM [37]:

$$V_{i,\text{opt}} \approx \mathbf{J}_d \mathbf{\Sigma}_{\boldsymbol{\xi},i} \mathbf{J}_d^T \tag{4.18}$$

where $\mathbf{J}_d \in \mathbb{R}^{1\times 7}$ is the first-order partial derivatives of the camera projection function with respect to the camera pose [98]. With the two-view consistent depth $D_{i,c}$ and depth variance $V_{i,c}$ maps, we update the Sim(3) pose constraints to refine the keyframe pose-graph.

4.4 Implementation

We use PyTorch [115] Multiprocessing[3] to implement the dense SLAM pipeline, enabling the depth prediction and dense mapping module to process in parallel. To improve computational performance, we process loops and deserialize ROS messages[4] using Boost.Python[5]. On average, the CNN depth prediction and optimization require 0.15 s and 0.2 s, respectively, to complete[6].

For semi-dense structure adaptive filter, we define a local window size as 5×5 with the following parameter settings: $\sigma_s = 76.5, \sigma_d = 2, \sigma_c = 0.3, \sigma_u = 2, \gamma = 0.0025$ and $\beta = 1.1$. Then, for densification of the semi-dense structure, we use PyTorch Autograd [116] with Adam optimizer [117] to minimize the cost function, where the learning rate is set to 0.05. To calculate the cost function, we set λ to 0.003 and the generalized Charbonnier function [114] parameters, ϵ and α, to 0.001 and 0.45, respectively. The number of optimization iterations is 30. Also, we resize the images and depth maps to 320×240 before performing depth prediction and densification. To perform pose-graph refinement, we set the error threshold $\tau_e = 0.001$ for obtaining two-view consistent depth regions.

In LSD-SLAM, we modify the `KFUsageWeight` and `KFDistWeight` parameters to 7.5. For the ICL/office0, ICL/living1, and TUM/seq2 sequences (see Table 4.1), we set `minUseGrad` parameter to 1. We set the frame-rate of all image sequences to 5 for better synchronization between the visualization of camera tracking and dense mapping. Though the increase of frame-rate, theoretically, should not affect the dense reconstruction accuracy except for the delayed visualization of the dense map because of the Multiprocessing implementation.

[3] `https://pytorch.org/docs/stable/multiprocessing.html`

[4] `http://wiki.ros.org/msg`

[5] `https://github.com/boostorg/python`

[6] The measurements are taken on a laptop computer equipped with an Intel 7820HK CPU and an Nvidia GTX 1070 GPU.

4.5 Experimental results and discussion

For the evaluation, we use ICL-NUIM [5] and TUM RGB-D [118] datasets, which contain ground truth depth maps and trajectories to measure the reconstruction accuracy. We follow the performance metrics used in [15]: The reconstruction accuracy is measured with the percentage of the depth values with relative errors of less than 10% and the pose estimation error is measured with the absolute trajectory error (ATE). Since our system does not produce metric scale reconstruction, and therefore each depth map needs to be scaled using the optimal trajectory scale (calculated with the TUM benchmark script[7]) and its corresponding Sim(3) scale for depth correctness evaluation.

The experimental results to validate our proposed method are organized as follows.

- Section 4.5.1 compares the dense reconstruction accuracy between our method and the state-of-the-art methods.
- Section 4.5.2 analyzes the effectiveness of the proposed adaptive filter.
- Section 4.5.3 provides an ablation study to compare different the effect of different cost functions, highlighting the better accuracy using our proposed cost function.
- Section 4.5.4 compares relative and absolute depth prediction accuracies to show the benefit of using relative depth prediction in dense reconstruction.
- Section 4.5.5 presents the improvement of camera pose estimation from our proposed pose-graph refinement.
- Section 4.5.6 analyzes the conditions required for accurate densification.

4.5.1 Reconstruction accuracy

Table 4.1 shows a comparison of the reconstruction accuracy between our method and the state-of-the-art dense SLAM systems (CNN-SLAM [15], Deep-

[7] `https://vision.in.tum.de/data/datasets/rgbd-dataset/tools`

Table 4.1: Comparison of overall reconstruction accuracy on the ICL-NUIM dataset [5] and the TUM RGB-D dataset [118]. (TUM/seq1: fr3_long_office_household, TUM/seq2: fr3_nostructure_texture_near_withloop, TUM/seq3: fr3_structure_texture_far.)

	Percentage of correct depth (%)					
Sequence	CNN-SLAM	DeepFactors*	DeepFusion	DeepFusion†(MiDaS)*	Ours (VNLNet)*	Ours (MiDaS)*
ICL/office0	19.410	**30.17**	21.090	15.934	17.395	17.132
ICL/office1	29.150	20.16	37.420	57.097	**60.909**	58.583
ICL/office2	37.226	-	30.180	**72.602**	68.914	72.527
ICL/living0	12.840	20.44	24.223	65.395	60.210	**65.710**
ICL/living1	13.038	20.86	14.001	75.631	69.980	**75.694**
ICL/living2	26.560	-	25.235	79.994	78.887	**80.172**
TUM/seq1	12.477	29.33	8.069	**69.990**	64.862	66.892
TUM/seq2	24.077	16.92	14.774	52.132	43.607	**59.744**
TUM/seq3	27.396	51.85	27.200	**76.433**	75.680	76.395
Average	22.464	27.10	22.466	62.801	60.049	**63.650**

*After aligned with ground truth scale

†Our implementation of DeepFusion

Fusion [89], and DeepFactors [112]): the first three columns show the reconstruction accuracy of the state-of-the-art systems and the last two columns show a comparison between using VNLNet (an absolute depth prediction CNN) and MiDaS (a relative depth prediction CNN) in our optimization framework (see Section 4.5.4). Because of the similarity of the optimization frameworks between our system and DeepFusion, we also include the results for running dense reconstruction with an additional CNN depth consistency error term in the cost function (labelled "†" in Table 4.1)[8]. Note that the reconstruction accuracy of our method is taken with an average of 5 runs. Our method outperforms the competitors except for the ICL/office0 sequence, as LSD-SLAM is unable to generate a good semi-dense structure under rotational motion, hence the degraded reconstruction performance in the densification of the semi-dense structure. The reconstruction results demonstrate the superiority of our system by comparing the last column with all other columns in Table 4.1. Figure 4.2 shows the use of our optimization framework to obtain more accurate densified depth maps from less accurate predicted relative depth maps.

[8]DeepFusion is not open-source, and therefore the results are based on the implementation of our optimization framework (see Section 4.4). Our implementation of the CNN depth consistency term is similar to that of DeepFusion except we use CNN depth for providing depth uncertainty (similar to Equation (4.11)).

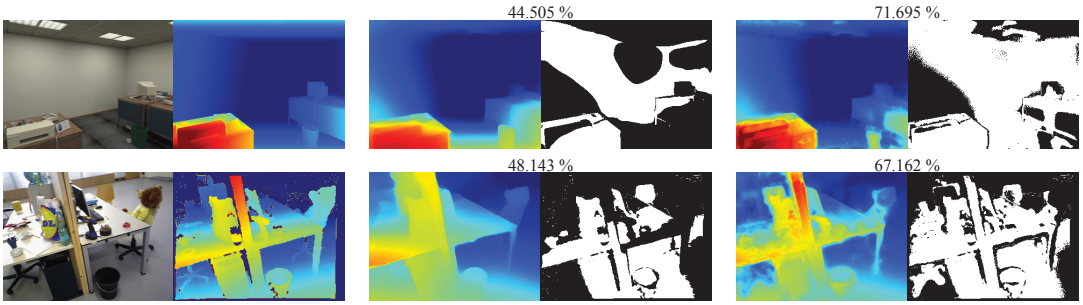

Figure 4.2: Demonstration of the effectiveness of our optimization framework by comparing the relative depth prediction accuracy from MiDaS before the densification with the densified depth map. (Left column) image and ground truth depth map. (Middle column) scale- and shift-corrected relative depth map and depth correctness mask. (Right column) densified depth map and depth correctness mask. The percentage of correct depth of the depth correctness mask is shown above.

Figure 4.3: The proposed adaptive filter on semi-dense depth map. From left to right: (back-projected) semi-dense depth map from LSD-SLAM, filtered semi-dense depth map, and keyframe image.

4.5.2 Results of the adaptive filter

We notice that the semi-dense structure from LSD-SLAM contains spurious map points, which may worsen the dense reconstruction performance. Figure 4.3 shows a qualitative comparison between the semi-dense depth maps by LSD-SLAM and the filtered depth maps, demonstrating the effectiveness of the adaptive filter in eliminating noisy depth pixels while preserving the structure of the scene. Quantitatively, the second row and the third row of Table 4.2 (labelled "f") shows about 5% improvement on using the adaptive filter in dense reconstruction (see also the last four rows).

Table 4.2: Effect of the error terms on the reconstruction accuracy. (TUM/seq1: fr3_long_office_household, $\circ$: our cost function, $\diamond$: simulated DeepFusion [89] cost function, †: not used in DeepFusion.)

	Percentage of correct depth (%)		
Energy term	ICL/living2	ICL/office2	TUM/seq1
1	62.620	57.563	55.031
1(c)	65.611	57.644	55.042
1(f)(c)	71.265	61.445	60.143
1(c)+2°	69.967	69.905	64.650
1(f)(c)+2°	**79.788**	**71.778**	67.319
1(c)+2+3$^\diamond$	70.167	69.863	64.730
1(f)(c)+2+3$^\diamond$	79.742	71.692	**67.323**

1. SLAM depth consistency
2. CNN depth gradient consistency
3. CNN depth consistency

(c). Generalized Charbonnier function†

(f). Adaptive semi-dense depth filtering†

4.5.3 Cost function analysis

Table 4.2 shows the reconstruction results using different combinations of error terms in the cost function. To ensure consistent measurement of the reconstruction accuracy using different cost functions, the keyframes—i.e., the semi-dense depth and depth variance maps, and the camera poses—are pre-saved so that the inconsistency between runs from LSD-SLAM does not influence the densification process. Consistent with the finding in DeepFusion, incorporation of CNN depth gradient consistency and CNN depth consistency improves the reconstruction accuracy dramatically, although our CNN does not explicitly predict depth gradient and depth gradient variance maps (see the second and last row). However, removing the CNN depth consistency term (the third and fourth last row), in our case, leads to better reconstruction accuracy (see also the third last and last column of Table 4.1); the added generalized Charbonnier function (the second row, labelled "c") also increases the accuracy.

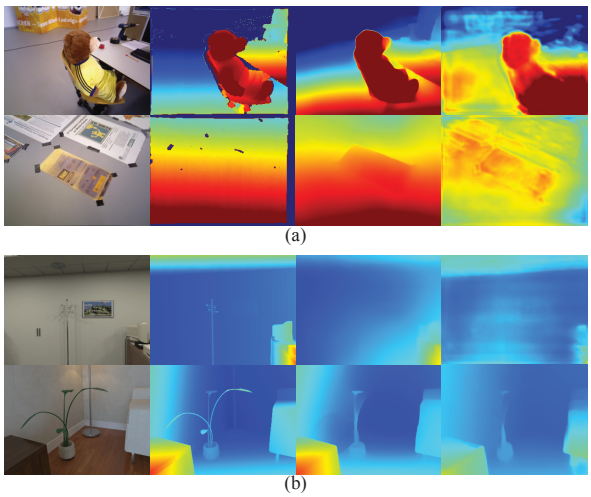

Figure 4.4: Qualitative comparison of relative depth maps from MiDaS and absolute depth maps from VNLNet on (a) the TUM RGB-D dataset and (b) the ICL-NUIM dataset. From left to right: image, ground truth depth map, depth prediction from MiDaS, and depth prediction from VNLNet.

4.5.4 Relative depth prediction vs. absolute depth prediction

To illustrate the advantage of using relative depth prediction CNNs (e.g., MiDaS), we perform the same densification step with an absolute depth prediction CNN, VNLNet[9] [66], and then compare the reconstruction accuracy between them. Neither MiDaS nor VNLNet has been trained on the TUM RGB-D and ICL-NUIM datasets to promote a fair comparison. In Table 4.1, we show that, in general, using scale- and shift-corrected relative depth prediction (labelled "MiDaS") instead of absolute depth prediction (other columns) has superior dense reconstruction performance, as a result of more accurate depth prediction from MiDaS than depth prediction from VNLNet (last and second last column of Table 4.3; see also Figure 4.4 for a qualitative comparison); Laina (second column of Table 4.3), another absolute depth prediction CNN being used in CNN-SLAM, is significantly less accurate than MiDaS, which indicates that the outperformance of our system may simply be because MiDaS provides more accurate depth prediction for densification.

[9]One crucial consideration in selecting a competing absolute depth prediction CNN is the runtime memory requirements. VNLNet is considered state-of-the-art at the time of experimental setup with a reasonable memory footprint.

Table 4.3: Comparison of depth prediction CNNs accuracy being used in CNN-SLAM (Laina [119]) and our system (VNLNet [66] and MiDaS [82]) on the ICL-NUIM dataset [5] and the TUM RGB-D dataset [118]. (TUM/seq1: fr3_long_office_household, TUM/seq2: fr3_nostructure_texture_near_withloop, TUM/seq3: fr3_structure_texture_far, abs: absolute depth prediction CNN, rel: relative depth prediction CNN.)

	Percentage of correct depth (%)		
Sequence	Laina (abs)	VNLNet (abs)*	MiDaS (rel)*
ICL/office0	**17.194**	11.791	13.059
ICL/office1	20.838	**45.866**	42.980
ICL/office2	30.639	**55.180**	55.136
ICL/living0	15.008	40.294	**54.287**
ICL/living1	11.449	55.806	**72.139**
ICL/living2	33.010	59.367	**67.130**
TUM/seq1	12.982	47.552	**54.860**
TUM/seq2	15.412	33.143	**55.136**
TUM/seq3	9.450	52.144	**57.255**
Average	18.452	44.571	**52.442**

*After scale- and shift-correction

Table 4.4: Comparison of absolute trajectory error on the ICL-NUIM dataset [5] and the TUM RGB-D dataset [118]. (TUM/seq1: fr3_long_office_household, TUM/seq2: fr3_nostructure_texture_near_withloop, TUM/seq3: fr3_structure_texture_far, abs: absolute depth prediction CNN, rel: relative depth prediction CNN, $^{\diamond}$: before pose-graph refinement, $^{\circ}$: after pose-graph refinement, *: (baseline) after pose-graph refinement with ground truth depth .)

	Absolute trajectory error (m)			
Sequence	CNN-SLAM	Ours$^{\diamond}$	Ours$^{\circ}$	Ours*
ICL/office0	**0.266**	0.352	0.295	0.260
ICL/office1	0.157	0.057	**0.046**	0.045
ICL/office2	0.213	0.159	**0.061**	0.045
ICL/living0	0.196	0.057	**0.039**	0.036
ICL/living1	0.059	**0.017**	0.018	0.017
ICL/living2	0.323	0.062	**0.059**	0.056
TUM/seq1	0.542	0.103	**0.075**	-
TUM/seq2	**0.243**	0.261	0.245	-
TUM/seq3	0.214	**0.108**	0.111	-
Average	0.246	0.131	**0.106**	-

-Not evaluated as not all the images have a corresponding depth map

4.5.5 Keyframe trajectory accuracy

Table 4.4 shows the camera tracking accuracy between our method and CNN-SLAM[10]. From the first two columns, we can see that our camera tracking performance, even without the pose-graph refinement, reduces the ATE of CNN-SLAM by almost 50%. Since both of the systems are built upon LSD-SLAM, the performance difference could be due to our configuration settings in LSD-SLAM (see Section 4.4). The last column shows a baseline performance of refining the pose-graph using ground truth depth to evaluate the effectiveness of pose-graph refinement. In general, pose-graph refinement reduces the ATE significantly to the extent that, in certain sequences, it is similar to that obtained by pose-graph refinement using the ground truth depth.

4.5.6 Conditions for accurate densification

In order to have accurate densification, both semi-dense depth maps and predicted depth maps from MiDaS have to be accurate. Statistically, we find that the aforementioned condition happens occasionally (see Table 4.5). Recall that 10 % relative error is used to evaluate depth accuracy. We define a semi-dense depth map from LSD-SLAM as *good* if more than 70 % of the semi-dense depth pixels have less than 10 % relative error; similarly, a scale- and shift-corrected dense depth map from MiDaS is considered *good* if more than 70 % of all depth pixels have less than 10 % relative error. As shown in the first row of Table 4.5, if both semi-dense depth and dense depth maps are of *good* quality, then the densified depth maps are also of *good* quality; in contrast, if both semi-dense and dense depth maps are of *bad* quality (see the last row of Table 4.5), then the probability of getting *good* densified depth maps is drastically reduced. In general, having *good* depth prior is beneficial for obtaining *good* densification results (see first and third rows of Table 4.5). Using the same metric, Figure 4.5 shows a qualitative comparison of back-projected depth maps (semi-dense depth maps, predicted depth maps, and densified depth maps).

[10]Only CNN-SLAM has the ATEs on the evaluation datasets.

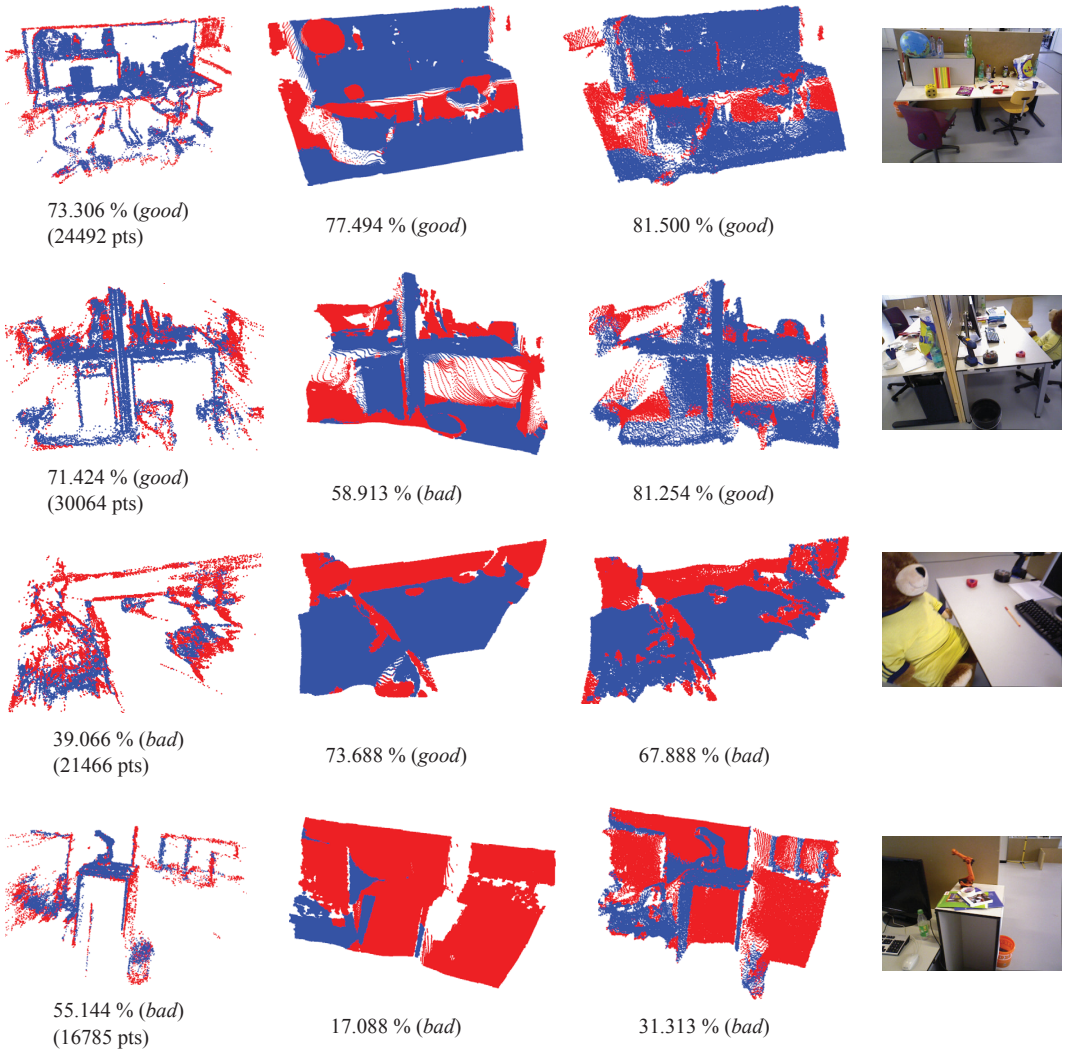

Figure 4.5: Back-projected point cloud generated from (left to right) LSD-SLAM [37], predicted depth map from MiDaS [82], and densified depth map using DeepRelativeFusion. Generally, fusing a *good* semi-dense depth map with a *good* predicted depth map results in a good densified depth map (see the first two rows). However, having either a *bad* semi-dense depth map or a *bad* predicted depth map is likely to generate a *bad* densified depth map (see the last two rows). Depth pixels that have less than 10 % relative error are in blue and are in red otherwise. The percentage of blue points is shown below their respective point clouds and the *good-ness* threshold is set to 70 %, as described in the text.

Table 4.5: Effect of semi-dense depth (from LSD-SLAM [37]) and predicted dense depth (from MiDaS [82]) accuracies on densification accuracy based on 142 keyframes generated on the TUM RGB-D fr3_long_office_household sequence [118].

Quality of depth			
Semi-dense	MiDaS	#keyframes	#*good* keyframes
good	*good*	14	14
good	*bad*	59	39
bad	*good*	13	12
bad	*bad*	56	9
	Total	142	74

A semi-dense depth map or dense depth map is defined as *good* if more than 70 % of the depth pixels have less than 10 % relative error; otherwise it is considered *bad*.

As mentioned in Section 4.3.1, we convert the MiDaS's relative depth to absolute depth by recovering the scale and shift factors using the semi-dense depth map in each keyframe. Despite being an efficient method of recovering absolute depth, as it only relies on a single view to perform the relative depth to absolute depth conversion, the accuracy of the conversion is strongly dependent on the depth distribution in the keyframe semi-dense depth map (see Figure 4.6). To maximize the densification accuracy, one may develop a more sophisticated scale- and shift-correction technique, e.g., leveraging multi-view consistency for better absolute depth alignment.

4.6 Qualitative reconstruction results on other datasets

Lastly, we present qualitative dense reconstructions on various datasets in Figure 4.7. One of the major bottlenecks of the state-of-the-art dense SLAM systems is the accurate depth prediction requirement in the testing scene. While the use of absolute depth prediction may help produce absolute scale reconstruction, it mostly makes sense in the context narrow application domain, such as dense scene reconstruction for self-driving cars. With the proposed use of relative depth prediction, we improve the versatility of our system by

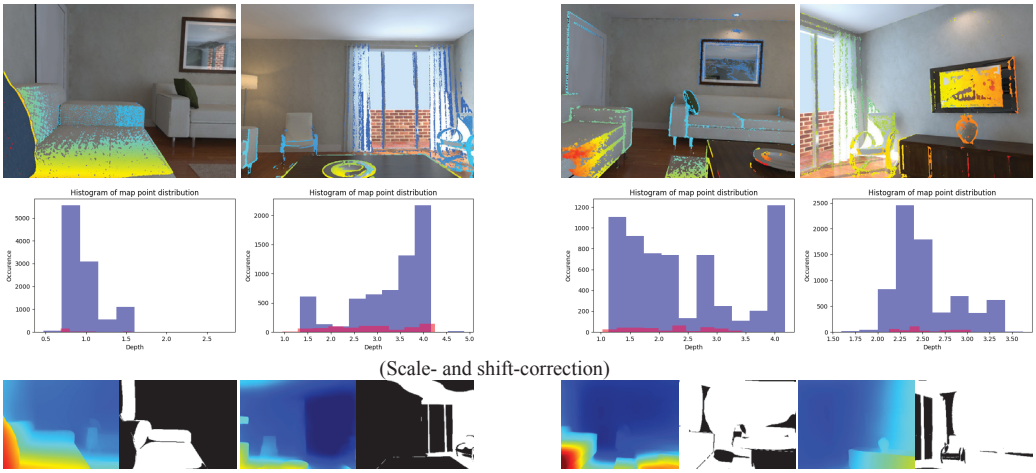

Figure 4.6: From top to bottom: colour-coded semi-dense depth map on the keyframe image (red is near and blue is far), histogram of the semi-dense depth distribution (depth values with less than 10 % relative error are shown in blue bars and are in red bars otherwise), and the scale- and shift-correct depth map with its correctness mask (depth regions that have less than 10 % relative error are shown in white). When performing scale- and shift-correction, using a partially distributed scene depth (see the histograms in the first two columns in which the depth count is skewed towards one end) results in poor recovery of absolute depth from predicted relative depth; whereas a well-distributed scene depth (see the histograms in the last two columns) leads to better recovery of absolute depth.

forgoing absolute scale reconstruction, which can be easily recovered using fiducial markers or objects with known scales. With accurate relative depth prediction as well as continuous expansion in single-image relative depth CNN training datasets, we are getting closer to solving dense monocular SLAM *in the wild*—dense scene reconstruction on arbitrary image sequences.

4.7 Summary

The key takeaways of this chapter are as follows.

- We propose a dense monocular SLAM system, named DeepRelativeFusion, capable of recovering a globally consistent 3D structure.
- Using a visual SLAM algorithm to reliably recover the camera poses and semi-dense depth maps of the keyframes, our proposed energy minimization framework exploits the accurate depth gradient information from single-image relative depth prediction as priors to densify the semi-dense depth maps.
- To improve the densification accuracy, we propose an adaptive filter for improving the semi-dense depth maps, a structure-preserving weighted average smoothing filter that considers the pixel intensity and depth of the neighbouring pixels.
- After densification, we update the keyframes with the optimized depth maps to improve pose-graph optimization, providing a feedback loop to refine the keyframe poses for accurate scene reconstruction.
- Our system outperforms the state-of-the-art dense SLAM systems quantitatively in dense reconstruction accuracy by a large margin.

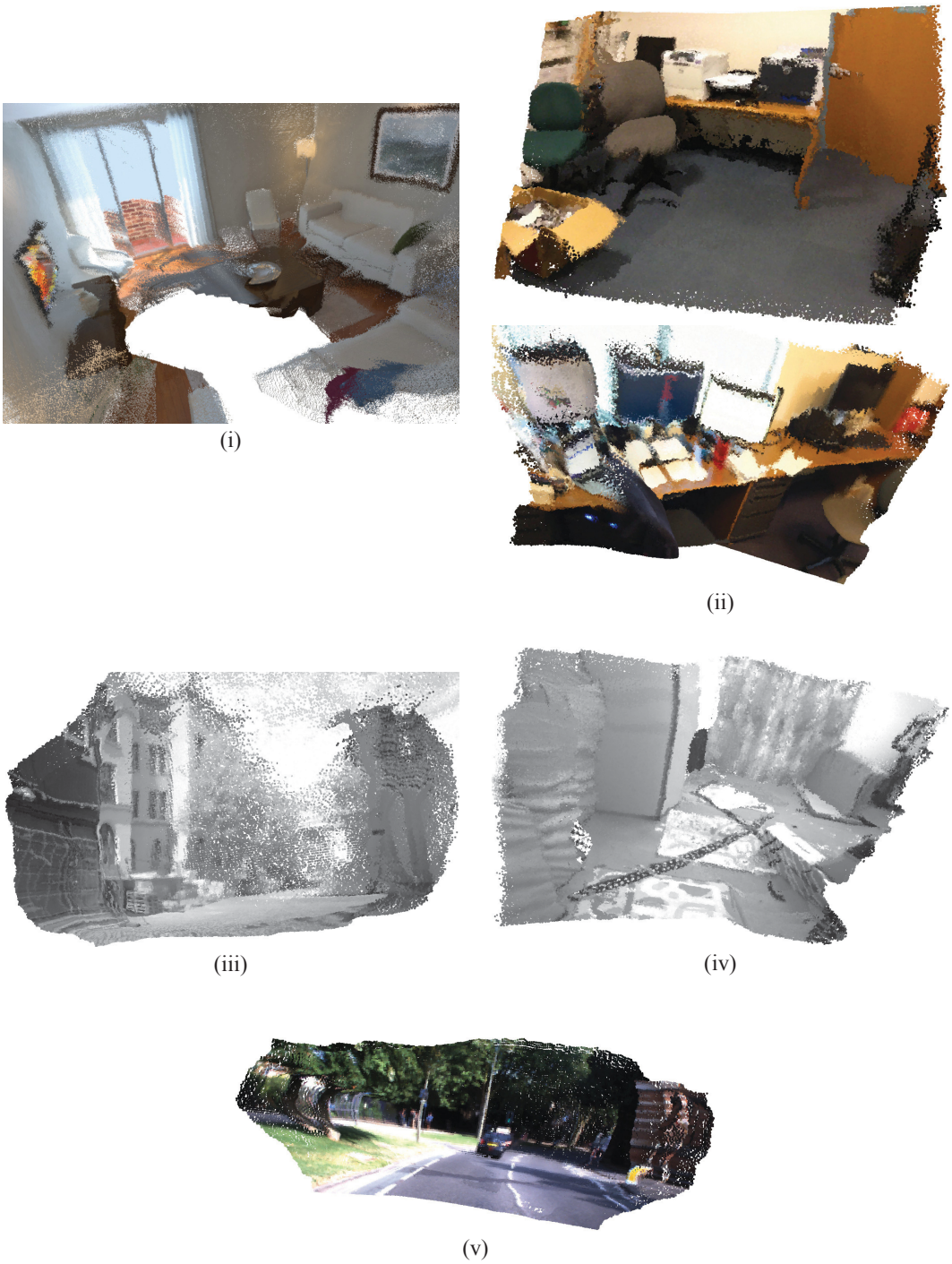

Figure 4.7: Qualitative reconstructions on various datasets. (i) ICL-NUIM [5] lr_kt2, (ii) ScanNet [19] scene0565_00 and scene0010_01, (iii) TUM MonoVO [120] Sequence_29, (iv) EuRoC MAV [121] v1_01 and (v) Oxford Robotcar 2014-06-22-15 [17].

Chapter 5

Online Mutual Adaptation of Deep Depth Prediction and Visual SLAM

5.1 Overview

In Chapter 3 and Chapter 4, we have demonstrated the advantages of using single-image depth prediction CNNs to solve data association and dense mapping problems. Specifically, we immediately observe degradation in CNN-SVO mapping (see Figure 5.1) and densification accuracy (see Section 4.5.4) with the use of less accurate depth prediction CNNs. This chapter is set out to answer the following question: Can we tune a depth prediction CNN with the help of a visual SLAM algorithm even if the CNN is not trained for the current operating environment and can the fine-tuned CNN in turn benefit the SLAM performance?

5.2 Related work

To overcome the domain gap of single-image depth prediction, online learning has been used to fine-tune a depth prediction CNN on the images available in the target domain. These images typically arrive sequentially [122]–[124]. One particular challenge for online fine-tuning is ensuring that learning does not overfit the most *recent* data, a problem known as *catastrophic forgetting* [92]. To mitigate *catastrophic forgetting*, *experience replay* has been

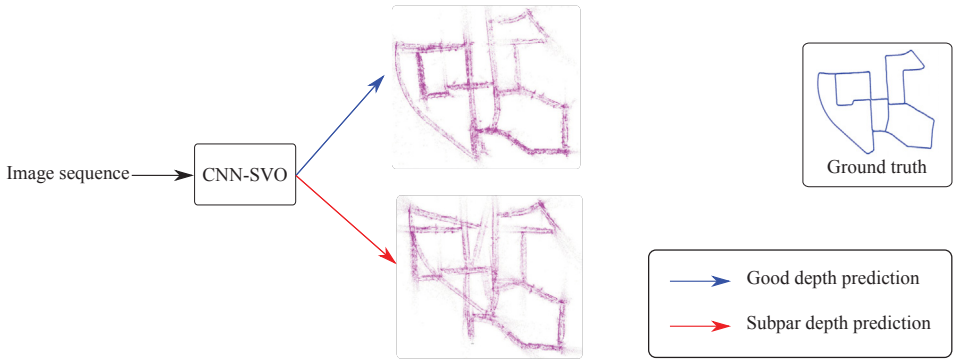

Figure 5.1: The decrease in SVO mapping performance using a lower quality depth prediction CNN.

proposed by inserting randomly sampled past training data into the current training batch [122], [123], [125]. An alternative solution is to regularize and preserve the CNN parameters that are important to the previously learned tasks [92], [126]. In a typical multi-task learning scenario, the importance of the CNN parameters can be measured by the magnitude of the gradients with respect to the loss function or output function through the training on a task [126]–[129], which intuitively determines how a perturbation in a CNN parameter affects the loss (see Appendix A for more information). Instead of measuring the parameter importance after learning a task and regularizing the parameters in learning the next task, Maltoni and Lomonaco [129] propose single-incremental-task (SIT) learning, which seeks to estimate and consolidate the parameter importance from batch to batch using synaptic intelligence (SI), so that the previously learned knowledge is retained throughout the learning. Unlike SIT, which has to deal with learning new instances or classes, our goal is to preserve the previously learned depth from batch to batch by consolidating the parameter importance using a semi-supervised loss, instead of supervised loss, which requires ground truth labels [129].

Depth prediction by a CNN has been widely used to benefit SLAM algorithms. A SLAM pipeline can be enhanced by improving feature matching [130] and photometric re-projection accuracies [12] through the use of depth prediction by CNNs. However, the predicted depth has to be accurate to achieve good performance, an assumption that is often violated in prac-

tical robotics applications. Therefore, instead of directly incorporating the predicted depth information into front-end tracking and mapping [12], [130], we design a VO pipeline that only incorporates online adapted depth information in optimizing the structure and motion estimation. To this end, we use two common procedures in SLAM optimization: map point culling and global BA. Removing noisy map points is an essential part of sparse SLAM for preventing erroneous state estimation from being included in optimization (see ORB-SLAM3 [58] and DSO [38]); on the other hand, global photometric BA [91], [131] has been shown to improve SLAM performance further. Assuming that online fine-tuning can obtain reasonable depth prediction (without extreme accuracy), we use depth prediction as a hint to cull the potential noisy map points from being included in global photometric BA, resulting in an improved SLAM accuracy. For implementation, we opt for SVO [51] for being computationally lightweight on an embedded computer, compared to other state-of-the-art sparse SLAM algorithms [38], [58].

5.3 Method

In this section, we outline our proposed online adaptation framework (see Figure 5.2). The framework consists of SLAM (Section 5.3.1), online CNN depth adaptation (Section 5.3.2), and global BA with adapted CNN depth (Section 5.3.3).

5.3.1 SLAM

We use SVO 2.0[1] (its monocular variant with edgelets) to generate a set of keyframes from a monocular image stream captured by the camera on a mobile robot. Associated with each keyframe are an image, a camera pose, and a set of sparse map points. While the keyframes are being generated, we publish two serialized channels[2] over a local network containing the newly created keyframes (image, pose and map points) and optimized camera poses and

[1] `https://github.com/HeYijia/svo_edgelet`

[2] The keyframes are converted to ROS messages before they are published. See `http://wiki.ros.org/msg` for more information.

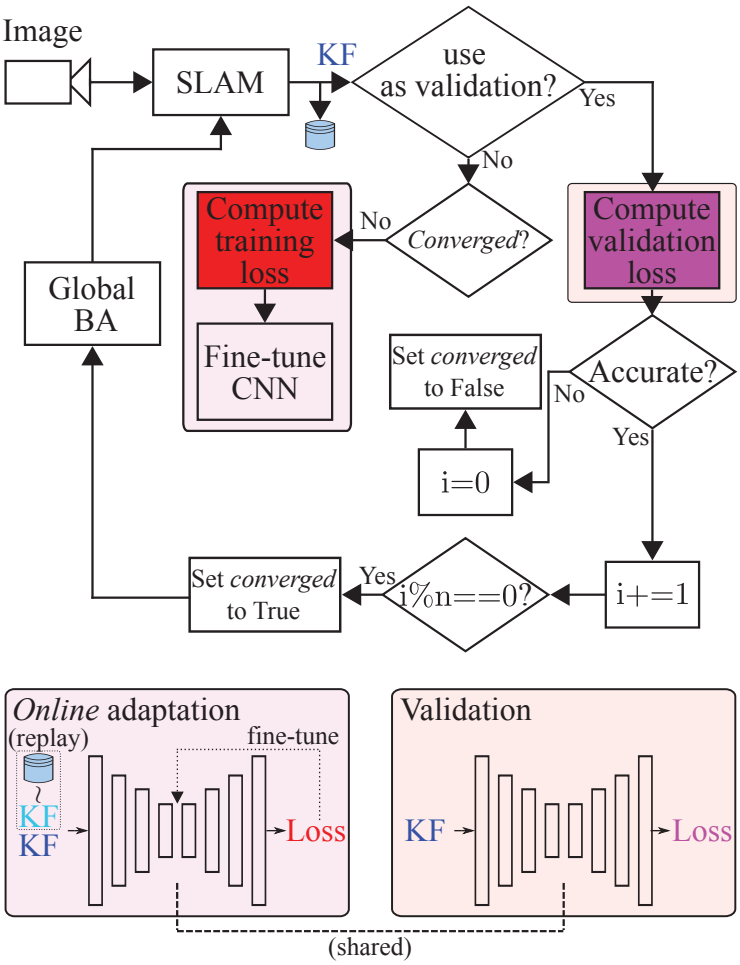

Figure 5.2: Our proposed online adaptation framework. We use a SLAM algorithm to generate a sequence of keyframes. The keyframes are classified as training or validation to fine-tune a depth prediction CNN and monitor the adaptation progress. If the training is not converged, we use the most recent keyframe and one randomly sampled old keyframe to fine-tune the CNN. Meanwhile, we calculate the validation loss once every m keyframes to determine if the predicted depth maps are accurate. We keep track of the number of continuous accurate depth predictions to perform global photometric BA if the CNN has been accurate for the past n keyframes. KF: keyframe.

map points of all keyframes. The reason for publishing the keyframe graph is that local BA, in most cases, improves the camera pose estimation. Therefore, it is beneficial for improving the training loss in the online adaptation.

5.3.2 Online CNN depth adaptation

Our online adaptation algorithm is inspired by *early stopping*[3] [132], whose goal is to avoid overfitting by stopping fine-tuning when depth prediction is reasonably accurate in terms of validation error (see Algorithm 1 for detailed steps). To this end, we validate the depth prediction accuracy once every m incoming keyframes to determine if the fine-tuning has converged. After determining the stopping criterion, we describe the training details of the online adaptation algorithm.

To adapt a depth prediction CNN on sequential data without *forgetting*, we improve upon the *experience replay* method (i.e., incorporating *recent* data and randomly selecting *old* data into the training batches [122], [123], [125]) by adding regularization. Training regularization requires the consideration of the learning of two separate tasks $\mathcal{T}_1$ (learn from *old* data) and $\mathcal{T}_2$ (learn from *recent* data). The goal is to maximize the training accuracy on both $\mathcal{T}_1$ and $\mathcal{T}_2$, ensuring that we preserve the depth prediction accuracy in the new environment. To reinforce the previously learned knowledge, we modify EWC [126] regularization in the following ways. First, instead of multi-task learning consolidation, we consolidate the importance of the parameters at each training batch and use it in the next batch, which considers the adaptation progress thus far to be absorbed into the posterior probability, i.e, the importance, of the parameters. Second, to estimate the posterior probability, we compute the gradient of the negative log-likelihood of the training loss, assuming constant observation noise for the input[4] [13], [133]. Let θ^* be the CNN parameters fine-tuned on the last training batch and θ be the current CNN parameters.

[3] `https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping`

[4] Realistically, observation noise can occur in the violation of brightness constancy assumption and motion blur. This issue can be mitigated by incorporating the heteroscedastic aleatoric uncertainty in training (studied in [13], [133]) and modifying a pre-trained depth prediction CNN to predict an additional uncertainty output.

Algorithm 1 online adaptation framework represented in a finite-state machine

```
state = IDLE
t = 0
n_converged = 0
m = 5                                          ▷ Validate every m steps
n = 3                                          ▷ patience in the context of EarlyStopping
τ_val = 0.2                                    ▷ Validation error threshold
while not quit do
    if state == IDLE then
        run_global_ba = false
        if t > 0 and t%m == 0 then
            Compute validation error (L_val)
            if L_val < τ_val then
                n_converged += 1
            else
                n_converged = 0
            end if
            if n_converged > 0 and n_converged%n == 0 then
                run_global_ba = true
            end if
        else
            state = FINE_TUNE
        end if
    else if state == FINE_TUNE then
        Fine-tune the depth prediction CNN
        state = IDLE
    end if
    if run_global_ba then
        go through all keyframe and publish the keyframe depth maps to trigger BA in SVO
    end if
    t += 1
end while
```

We introduce an additional regularization term to the training loss[5] [126]:

$$\mathcal{L}_{\text{EWC}} = \mathcal{L}_{\text{train}} + \sum_i \frac{\beta}{2} \hat{\mathbf{F}}_i (\theta_i - \theta_i^*)^2 \tag{5.1}$$

with

$$\begin{aligned} \mathbf{F} &= \mathbf{F} + \mathbf{F}^{(j)} \\ \hat{\mathbf{F}} &= \max(\frac{\mathbf{F}}{j}, \max_{\mathbf{F}}), \end{aligned} \tag{5.2}$$

where j is the number of trained batches and $\mathbf{F}$ the consolidated empirical Fisher information matrix at the end of every training batch. $\hat{\mathbf{F}}_i$ is importance of the i-th parameter, which is obtained from the averaged parameter importance matrix $\hat{\mathbf{F}}$ to ensure the preservation of the previously learned knowledge. The empirical Fisher information matrix converges to the Fisher information matrix as we incorporate more training samples [134]. $\max_{\mathbf{F}}$ determines the maximum value of $\hat{\mathbf{F}}$.

To compute the training loss ($\mathcal{L}_{\text{train}}$), we use a *semi-supervised* training loss [73], [103], which consists of photometric re-projection ($\mathcal{L}_{\text{photo}}$), sparse depth ($\mathcal{L}_{\text{sparse\_depth}}$) and smoothness loss ($\mathcal{L}_{\text{smooth}}$) terms, to be defined below.

First, the photometric re-projection error $\mathcal{L}_{\text{photo}}$ is given by:

$$\mathcal{L}_{\text{photo}} = \frac{1}{|\Omega|} \sum_{\mathbf{p} \in \Omega} \min_s pe(I_i, I_{s \to i}, \mathbf{p}), \text{ and} \tag{5.3}$$

$$\begin{aligned} pe(I_i, I_{s \to i}, \mathbf{p}) = \frac{\alpha}{2} &(1 - \text{SSIM}(I_i, I_{s \to i})(\mathbf{p}) \\ &+ (1 - \alpha) \parallel I_i(\mathbf{p}) - I_{s \to i}(\mathbf{p}) \parallel_1, \end{aligned} \tag{5.4}$$

where $pe(\cdot, \cdot, \cdot)$ calculates the minimum per-pixel photometric re-projection error [103], $s \in \{i-1, i+1\}$ the indices of the adjacent keyframes and Ω the set of pixel locations in I_i. To reconstruct the target image from an adjacent image $I_{s \to i}$, we use bilinear sampling at the pixel locations (Ω_s) that can be warped to the adjacent image using depth prediction $D_{i,\text{CNN}}$ and relative keyframe pose transformation $\mathbf{T}_{i \to s}$ from SVO:

$$\begin{aligned} I_{s \to i}(\mathbf{p}) &= I_s(\mathbf{p}') \quad \forall \mathbf{p} \in \Omega_s, \\ \mathbf{p}' &= \pi(\mathbf{T}_{j \to i} D_{i,\text{CNN}}(\mathbf{p}) \pi^{-1}(\mathbf{p})), \end{aligned} \tag{5.5}$$

[5] We explain the details in Appendix A.

where $\mathbf{p}'$ is a re-projected pixel location.

Then, to define $\mathcal{L}_{\text{sparse\_depth}}$, we project the sparse map points generated by SVO in each keyframe to form sparse depth maps as *ground truth* labels [73], [135]:

$$\mathcal{L}_{\text{sparse\_depth}} = \frac{1}{|\Omega_{i,\text{sparse}}|} \sum_{\mathbf{p} \in \Omega_{i,\text{sparse}}} |\frac{1}{D_{i,\text{CNN}}(\mathbf{p})} - \frac{1}{D_{i,\text{sparse}}(\mathbf{p})}|, \tag{5.6}$$

where $D_{i,\text{sparse}}$ is the sparse depth map of keyframe i, $\Omega_{i,\text{sparse}}$ the set of re-projected (sub)pixel locations containing depth values of their corresponding map points from SVO, and $|\Omega_{i,\text{sparse}}|$ the number of valid re-projections. By using inverse depth, near depth is penalized more heavily than far depth.

Finally, we compute the edge-aware smoothness loss $\mathcal{L}_{\text{smooth}}$ using the image gradient:

$$\begin{aligned} \mathcal{L}_{\text{smooth}} = \frac{1}{|\Omega|} \sum_{\mathbf{p} \in \Omega} & |\partial_x D_{i,\text{CNN}}(\mathbf{p})| e^{-|\partial_x I_i(\mathbf{p})|} \\ & + |\partial_y D_{i,\text{CNN}}(\mathbf{p})| e^{-|\partial_y I_i(\mathbf{p})|} \end{aligned} \tag{5.7}$$

where ∂ is the gradient operator.

Combining the three loss terms, the final training loss is

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{photo}} + \lambda_1 \mathcal{L}_{\text{sparse\_depth}} + \lambda_2 \mathcal{L}_{\text{smooth}}, \tag{5.8}$$

where λ_1 and λ_2 are the weighting for the sparse depth loss and smoothness loss, respectively. Algorithm 2 details the steps involved in the online fine-tuning. To validate the adaptation progress, we use the sparse depth loss term on all validation keyframes as the validation loss:

$$\mathcal{L}_{\text{val}} = \mathcal{L}_{\text{sparse\_depth}}. \tag{5.9}$$

5.3.3 Global BA

To optimize the structure and motion, we employ the traditional photometric BA [91], which jointly optimizes the keyframe poses and map points. However, we found that the map points generated by SVO, albeit the sparsest amongst the state-of-the-art visual SLAM algorithms [38], [58], still contain noisy and

Algorithm 2 Online fine-tuning with parameter importance regularization

Init θ from a pre-trained depth prediction CNN
$\hat{F} = 0$ ▷ parameter importance matrix
while not quit **do**
  if `state` == `IDLE` **then**
    continue ▷ Low validation error; skip training
  end if
  $\theta_{\text{cpy}} = \theta$ ▷ Make a copy
  Get latest keyframe $\mathcal{K}_i$ from SVO
  Train the CNN on $\mathcal{K}_i$ and $\mathcal{K}_{j\sim\{0,\ldots,(i-1)\}}$ with $\mathcal{L}_{\text{EWC}}$ (see Eq. 5.1) according to θ and $\hat{F}$
  Consolidate the parameter importance matrix $\hat{F}$ according to θ_{cpy} and θ (see Eq. 5.2)
end while

redundant map points. Therefore, with the learned depth information, we introduce a map point culling step before performing photometric BA. To determine if a map point should be culled, we identify a host keyframe for the map point (see Figure 5.3) and check if the depth of the map point is within the *correctness* range:

$$g(d_{\text{mp}}, d_{\text{CNN}}) = \begin{cases} 1, & \text{if } |d_{\text{mp}} - d_{\text{CNN}}| < \alpha d_{\text{CNN}} \text{ or} \\ & \quad d_{\text{CNN}} > d_{\max} \\ 0, & \text{otherwise} \end{cases} \tag{5.10}$$

where d_{CNN} and d_{mp} are the predicted CNN depth and the depth of the map point in the host keyframe, $d_{\max}$ a threshold that defines an *effective* depth range of the CNN depth values (similar to a depth sensor) to avoid far map points from being culled prematurely. $g(\cdot)$ evaluates if the map point is valid. Then, we use the standard photometric BA formulation [91] to globally optimize the structure and camera poses (collectively defined as the SLAM variables $\mathcal{X}$) given by:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \frac{1}{2} \sum_k \|\mathbf{e}_k\|_2^2 \tag{5.11}$$

with

$$\mathbf{e}_k = \mathbf{z}_k - h_k(\mathcal{X}_k) \tag{5.12}$$

where $\mathbf{e}_k$ is the photometric error induced by a subset of the SLAM variables $\mathcal{X}_k \subseteq \mathcal{X}$, $\mathbf{z}_k$ the reference image patch and $h_k(\cdot)$ the measurement model to

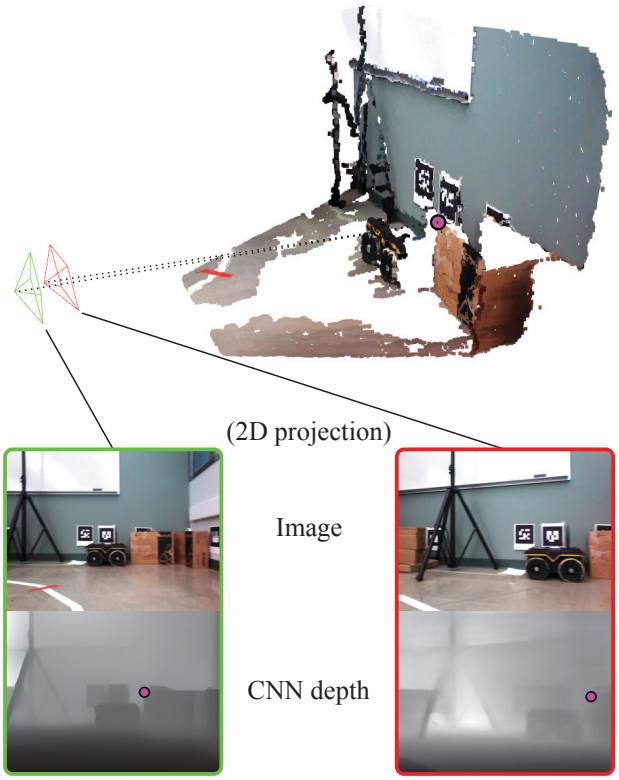

Figure 5.3: Assuming the magenta map point is observed in two keyframes (the red and green camera frustums), a host keyframe is selected based on the validation loss ($\mathcal{L}_{\text{val}}$) of the predicted CNN depth, and in this case, the green keyframe has a lower $\mathcal{L}_{\text{val}}$ and hence is being selected as the host keyframe of the magenta map point.

obtain the re-projected image patch.

5.4 Evaluation

To evaluate the performance of our proposed method for online CNN adaptation experimentally, our mobile robot hardware setup consists of the following main components: a TurtleBot, an Nvidia Jetson AGX Xavier, an Orbbec Astra RGBD camera, and a laptop[6], as shown in Figure 5.4. With the Jetson and laptop connected to the same wireless network, we run the SLAM process (tracking, mapping, and BA) on the Jetson and the fine-tuning process on the laptop.

To illustrate the effectiveness of the online adaptation, we use Monodepth2's

[6]Specifications: Intel 7820HK CPU and Nvidia GTX 1070 GPU

`mono+stereo_640x192` CNN model, which has been pre-trained on outdoor scenes fine-tuned in an indoor environment. For the network to learn, we use Adam optimizer with a learning rate of 10^{-3}, and set the weights of the loss function λ_1, λ_2 and β at 0.1, 0.1 and 5×10^7, respectively. For EWC consolidation of parameter importance matrix, $\max_F$ is set to 0.001.

In SVO, the main change we made is the increased number of tracked features in the keyframes. We have modified the following parameter values: `max_fts` = 500, `grid_size` = 20, and `core_kfs` = 5. For performing map point culling, we define the *correctness* range by setting $\alpha = 0.5$ and $d_{\max} = 1.5$. This particular design allows for more map points to be generated for fine-tuning the depth prediction CNN. The online adapted depth is then used for removing the potential noisy map points. After noisy point removal, each map point has a maximum of five photometric re-projections to the nearby keyframes, including the existing 3D-2D constraints (i.e., the list of observed keyframes for the map point), for performing global photometric BA in a separate thread. We get a 3×3 image patch around the re-projected image coordinates for each re-projection edge to compute the photometric errors[7].

In the following, we present experimental results to validate the performance of our proposed method. Section 5.4.1 details the dataset that we collected in our laboratory and its purposes. Section 5.4.2 compares and contrasts our proposed online adaptation method with regularization against the state-of-the-art methods. Section 5.4.3 conducts an ablation study to compare different online adaptation schemes in overcoming *catastrophic forgetting*. Section 5.4.4 presents quantitative and qualitative results of our proposed map point culling with online adapted depth and global photometric BA to improve map reconstruction accuracy. Section 5.4.5 analyzes the accuracy of online adapted and pre-trained depth prediction to show the advantage of using online adaptation to improve SLAM performance. And lastly, Section 5.4.6 evaluates the runtimes of our proposed online adaptation framework.

[7]More details can be found in Appendix-B1

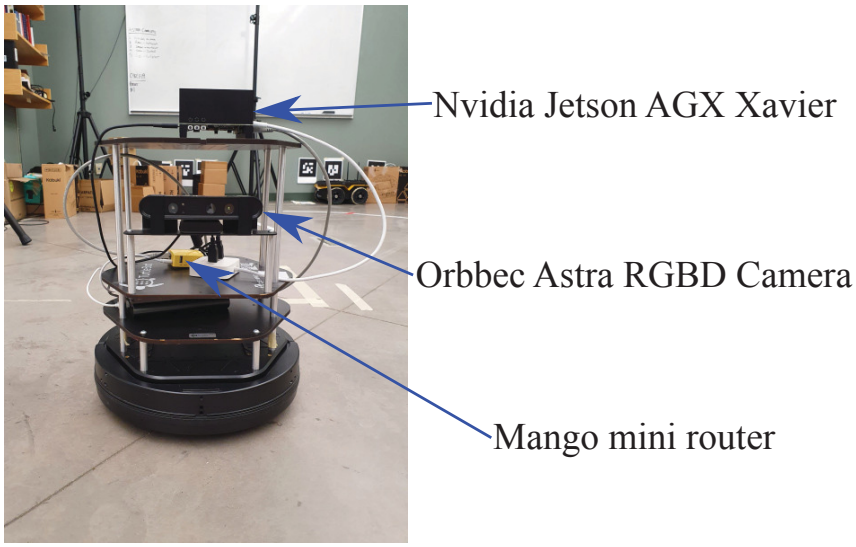

Figure 5.4: A TurtleBot equipped with an Nvidia Jetson AGX Xavier and an Orbbec Astra RGBD camera. A Mango mini router is used to create a local wireless network to communicate between the Jetson and a laptop.

5.4.1 Laboratory dataset

The reason for collecting our own dataset is that the image sequences in existing benchmarking datasets [18], [19], [118] are not long enough to evaluate and illustrate the effectiveness of our proposed online adaptation. Our dataset contains two sequences (dubbed `Lab1` and `Lab2`) that we collected in our laboratory (see Figure 5.6 for the example images of the experimental environment). We record both sequences in the same environment, with the difference that the `Lab2` sequence (16366 images) is longer than the `Lab1` sequence (10611 images).

5.4.2 Online adaptation

To evaluate the effectiveness of our proposed online adaptation, we compare our method against the state-of-the-art methods: a SLAM-based approach by Luo et al. [124] (Section 5.4.2) and a learning-based approach by Kuznietsov et al. [123] (Section 5.4.2). For the comparison, we adopt the datasets and performance metric in [124]. The datasets are the ICL-NUIM [5] and TUM RGB-D [118] datasets and the performance metric is the percentage of overall depth pixels below 10% relative errors, i.e., $\frac{|D-D_{gt}|}{D_{gt}} < 0.1$.

Table 5.1: A comparison between the overall depth accuracy of our method and Luo et al.'s [124] SLAM-based online adaptation on the ICL-NUM [5] and TUM RGB-D [118] datasets. (TUM/seq1: fr3_long_office_household, TUM/seq2: fr3_nostructure_texture_near_withloop, TUM/seq3: fr3_structure_texture_far)

	Percentage of correct depth (%)			
Sequence	Pre-trained [124]	[124]	Pre-trained	Ours
ICL/office0	19.117	**22.206**	8.766	12.541
ICL/office1	28.086	**31.289**	19.739	21.870
ICL/office2	21.695	21.695	4.591	**42.244**
ICL/living0	18.680	23.278	7.726	**41.375**
ICL/living1	21.071	22.774	8.518	**49.075**
ICL/living2	16.150	20.995	13.509	**25.159**
TUM/seq1	18.208	20.259	10.999	**23.861**
TUM/seq2	25.796	29.014	12.678	**52.162**
TUM/seq3	20.668	30.156	10.295	**37.848**
Average	21.052	24.630	10.758	**34.015**

SLAM-based online adaptation

Table 5.1 compares the overall depth accuracy between our method and a similar method by Luo et al. [124]. Our method improves the overall depth accuracy by around 23% (the last two columns), compared to 4% (2nd and 3rd column) by Luo et al. [124]. The performance gap could be due to the number of keyframes used in the fine-tuning; we use all the keyframes generated by SVO [51], whereas Luo et al. [124] use the keyframe pairs by LSD-SLAM [37] that have a dominant horizontal motion as simulated static stereo pairs. Besides, Luo et al. [124] perform online adaptation using most *recent* keyframes only, which performs the worst in the online adaptation scheme comparison (see Section 5.4.3). Note that Luo et al. [124] evaluate the keyframe depth accuracy based on the fine-tuned CNN models in different fine-tuning stages, which may result in different overall depth correctness should the final fine-tuned CNN model be used in the evaluation.

Table 5.2: A comparison between the overall depth accuracy of our method and CoMoDa's [123] end-to-end online adaptation on the ICL-NUM [5] and TUM RGB-D [118] datasets. Both our method and CoMoDa [123] are fine-tuned on the same pre-trained CNN model, `mono+stereo_640x192` [103]. (TUM/seq1: fr3_long_office_household, TUM/seq2: fr3_nostructure_texture_near_withloop, TUM/seq3: fr3_structure_texture_far)

	Percentage of correct depth (%)		
Sequence	Pre-trained*	Ours*	CoMoDa* [123]
ICL/office0	13.660	**32.98**	21.054
ICL/office1	22.453	38.383	**41.623**
ICL/office2	22.514	**44.790**	37.374
ICL/living0	17.659	**48.158**	35.164
ICL/living1	22.585	**51.531**	46.148
ICL/living2	21.606	**35.757**	35.571
TUM/seq1	16.931	27.915	**33.120**
TUM/seq2	17.722	**41.782**	33.708
TUM/seq3	23.026	**50.339**	39.769
Average	19.795	**41.293**	35.948

*Median scaling to ground truth depth

Learning-based online adaptation

Table 5.2 compares our method against a learning-based online adaptation method, CoMoDa[8]. Overall, our proposed SLAM-based method outperforms CoMoDa by around 5% (the last two columns). One particular challenge in end-to-end online adaptation is the simultaneous fine-tuning of depth and pose prediction CNNs, and the pose prediction CNN is not trained in the indoor test environments; on the contrary, we obtain camera poses from SVO [51]. Furthermore, accurate camera pose estimation is essential when performing online adaptation (see Equation 5.5). Plus, CoMoDa does not use regularization in online adaptation, which is a reason that it performs worse than our method (see Section 5.4.3).

[8]We generate results using CoMoDa's [123] open-source code (`https://github.com/Yevkuzn/CoMoDA`)

5.4.3 Learning against *catastrophic forgetting* in online adaptation

To evaluate the effectiveness of our proposed regularization in online adaptation, we perform an ablation study to measure the impact of different adaptation schemes on alleviating *catastrophic forgetting*. To this end, we examine the overall depth accuracy (using the same 10% relative error as the metric) on the ICL-NUIM [5] and TUM RGB-D [118] datasets.

Table 5.3 shows that performing online adaptation using only the most *recent* keyframes has the worst overall depth accuracy (3rd column), compared to using the most *recent* keyframe with *experience replay* and regularization (the last two columns). Moreover, Figure 5.5 illustrates *catastrophic forgetting* on the `lr_kt1` sequence, where the CNN is overfitted to the most *recent* keyframes in the sequence. In general, a combination of *experience replay* and regularization in online adaptation has the best overall depth accuracy (see the last column in Table 5.3 and Figure 5.5). Thus, it validates the effectiveness of our proposed online adaptation scheme. However, for a short sequence, regularization can inhibit changes in the CNN parameters and hurt the online adaptation accuracy (2nd last row in Table 5.3). For the sake of completeness, we compare the online adaptation performances using the three popular regularization techniques[9]: EWC [126], MAS [127] and SI [128], and they perform similarly.

5.4.4 Effect of global photometric BA with map point culling on SLAM accuracy

After the depth prediction CNN has been fine-tuned to a reasonable accuracy, the next step of our proposed method uses the online adapted depth to remove potential noisy map points and then performs global photometric BA to improve SLAM performance.

To measure the camera tracking performance before and after the global photometric BA, we use the standard absolute trajectory RMSE (ATE) as the

[9]See Appendix A.2 for more information on the SI and MAS regularizations. The regularization strength (λ in Eq. A.6) for SI and MAS are set to 1.0 and 0.5, respectively.

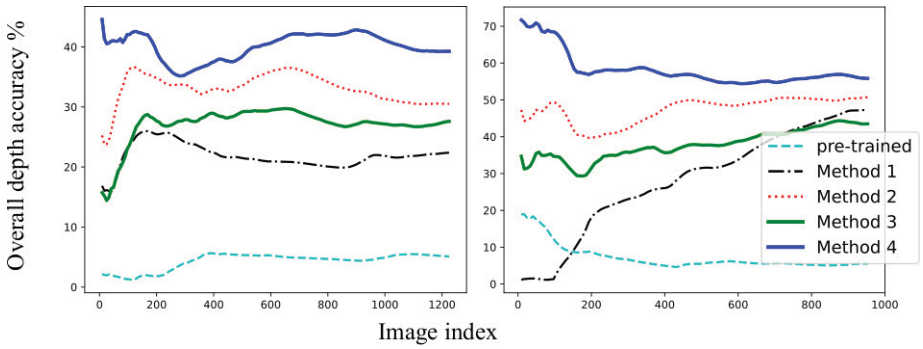

Figure 5.5: A comparison of different online adaptation schemes tested on the ICL-NUIM [5] `of_kt3` (left) and `lr_kt1` (right) using the final online adapted CNN. Adaptation accuracy is measured by the averaged percentage of overall depth accuracy over all frames up to the frame. (Method 1: fine-tuning on most *recent* keyframes only; Method 2: fine-tuning on the most *recent* keyframe with *experience replay*; Method 3: fine-tuning on the most *recent* keyframe with regularization; Method 4: fine-tuning on the most *recent* keyframe with *experience replay* and regularization.

Table 5.3: An ablation study on different online adaptation schemes using our proposed framework on the ICL-NUM [5] and TUM RGB-D [118] datasets. (TUM/seq1: fr3_long_office_household, TUM/seq2: fr3_nostructure_texture_near_withloop, TUM/seq3: fr3_structure_texture_far)

	Percentage of correct depth (%)			
Sequence	Pre-trained	Ours(a)	Ours(b)	Ours(c)
ICL/office0	8.766	7.249	**12.734**	12.541
ICL/office1	19.739	4.034	**22.613**	21.870
ICL/office2	4.591	21.449	34.609	**42.244**
ICL/living0	7.726	13.179	22.883	**41.375**
ICL/living1	8.518	38.246	41.149	**49.075**
ICL/living2	13.509	9.110	13.117	**25.159**
TUM/seq1	10.999	16.012	19.782	**23.861**
TUM/seq2	12.678	14.457	37.539	**52.162**
TUM/seq3	10.295	49.515	**51.545**	37.848
Average	10.758	19.250	28.441	**34.015**

(a) Online adaptation with two most *recent* keyframes
(b) Online adaptation with the most *recent* keyframe and *experience replay*
(c) Online adaptation with the most *recent* keyframe, *experience replay* and regularization

Table 5.4: A comparison of the camera tracking ATEs with and without global photometric BA on the ICL-NUM [5] and TUM RGB-D [118] datasets. (TUM/seq1: fr3_long_office_household, TUM/seq2: fr3_nostructure_texture_near_withloop, TUM/seq3: fr3_structure_texture_far)

Sequence	Without BA*	With BA*
ICL/office0	0.363	**0.323**
ICL/office1	**0.196**	0.204
ICL/office2	**0.170**	0.171
ICL/living0	**0.220**	0.227
ICL/living1	0.032	**0.026**
ICL/living2	0.161	**0.158**
TUM/seq1	0.088	**0.073**
TUM/seq2	0.029	**0.011**
TUM/seq3	**0.011**	0.012
Average	0.141	**0.134**

*Aligned to ground truth trajectory

performance metric on the ICL-NUIM [5] and TUM RGB-D [118] datasets, which contains ground truth camera poses for the evaluation. Table 5.4 compares the ATEs obtained from SVO and SVO with our proposed map point culling and global photometric BA. In general, performing global photometric BA leads to a lower overall ATE than that without the global BA (compare the last two columns in Table 5.4). Our proposed global photometric BA with map point culling reduces the ATE by more than 50% (from 0.029 to 0.011) on the `TUM/seq2` sequence, likely due to the improved depth prediction with online adaptation (check the corresponding rows in Table 5.3).

To highlight the effectiveness of the online adaptation, we evaluate the accuracy of the SVO map on long sequences, `Lab1` and `Lab2`, from our Laboratory dataset. We use the scale-invariant inverse depth error as the performance metric [70] given by:

$$e_{\text{si}} = \sqrt{\frac{1}{N}\sum_i d_i^2 - \frac{1}{N^2}(\sum_i d_i)^2} \tag{5.13}$$

where $d_i = \log z_i - \log z_i^*$, and the superscript * indicates the ground truth depth.

To allow for a more accurate online adaptation accuracy, we increase the

Table 5.5: A comparison of the number of points and depth error of the SVO map with and without our proposed map point culling and global photometric BA. A larger `max_fts` generates more map points in SVO.

BA	MP culling	SVO param (`max_fts`)	`Lab1`		`Lab2`	
			No. map points	e_{si}	No. map points	e_{si}
		120	13992	0.533	19856	0.607
✓		120	14469	0.545	20464	0.625
✓	✓	120	7435	0.473	9226	0.495
		500	19179	0.527	25743	0.540
✓		500	19208	0.515	23998	0.528
✓	✓	500	9916	0.493	12686	0.472

Note: `max_fts` is set at 120 by default in SVO.

number of map points by tweaking the `max_fts` parameter in SVO. Table 5.5 shows that the SVO map accuracy is improved by our proposed map point culling with online adapted depth prediction and global photometric BA (compare the e_{si} with and without BA in Table 5.5 and Figure 5.6(a) and 5.6(b)). The improved map accuracy also comes at the expense of removing good map points, which can be mitigated by having a training scheme with better online adaptation performance. As a result, our proposed online adaptation framework achieves a mutual adaptation of depth prediction and visual SLAM.

5.4.5 Online adaptation vs. relative depth prediction

A competing idea to our online adaption for resolving the domain gap in single-image depth prediction is to train a network with a large number of datasets across multiple domains to improve its generalization. Unlike absolute depth prediction, which is trained in a narrow domain, state-of-the-art relative depth prediction CNN models, such as MiDaS [82] and DiverseDepth [83], have been trained on an extensive collection of datasets. Therefore, to compare between online adapted depth and pre-trained depth prediction, we use the same scale-invariant inverse depth error e_{si} (see Equation 5.13).

Table 5.6 shows a quantitative comparison of the scale-invariant depth errors of sparse SVO map points, MiDaS, DiverseDepth and our fine-tuned Monodepth2 CNN model. On both sequences (2nd and 3rd column), our method

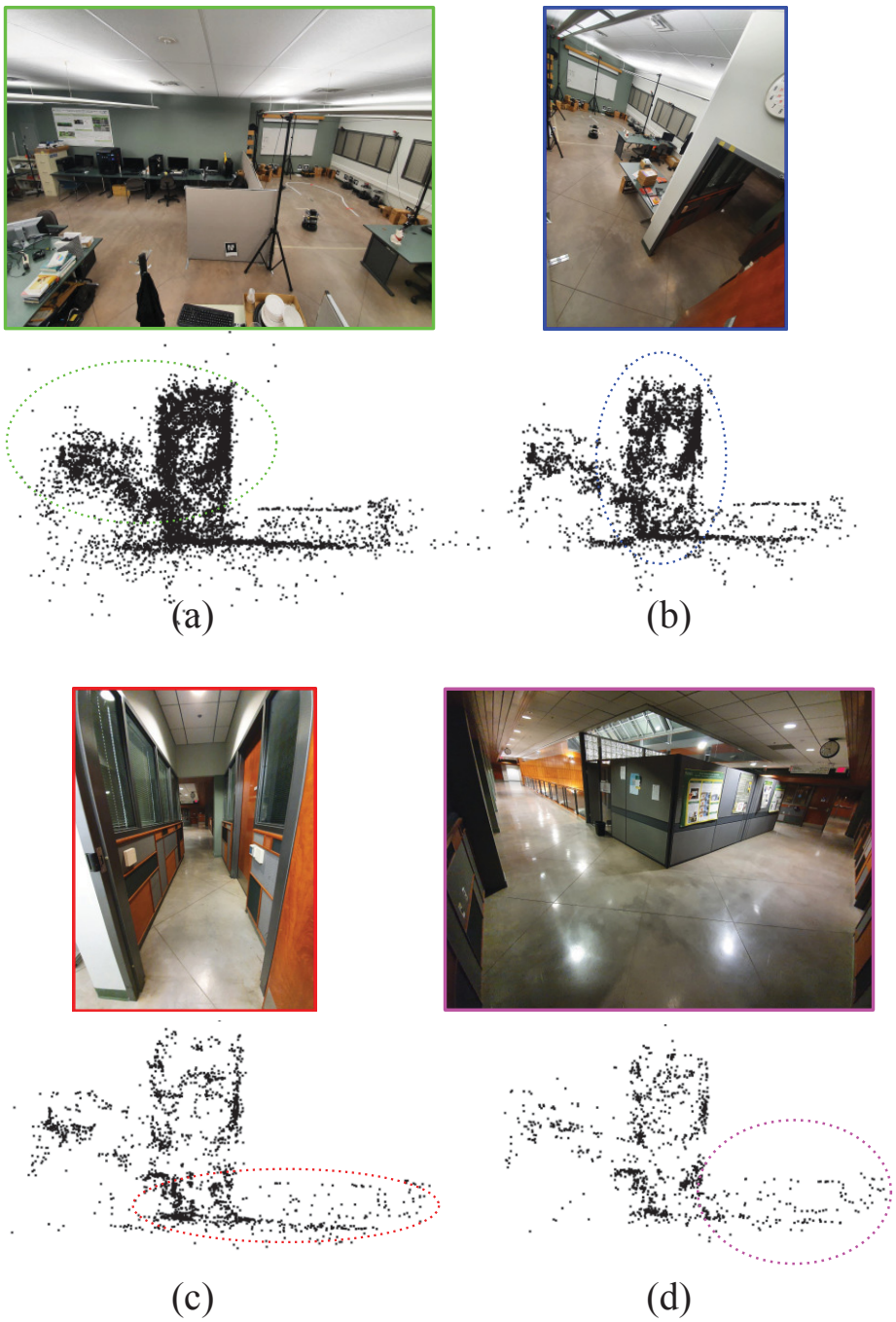

Figure 5.6: Qualitative comparisons between different *correctness* thresholds used in map point (MP) culling: (a) no culling, (b) MP culling with $\alpha = 0.5$, (c) MP culling with $\alpha = 0.25$ and (d) MP culling with $\alpha = 0.15$.

Table 5.6: A comparison of scale-invariant depth errors of SVO [51] map points, MiDaS [82], DiverseDepth [83] and our online adapted Monodepth2 [103] CNN model.

Method	e_{si}	
	`Lab1`	`Lab2`
MiDaS (`V2.1`)	0.271	0.256
MiDaS (`V3.0`)	0.315	0.322
DiverseDepth	0.450	0.435
Pre-trained	0.401	0.399
Ours	**0.226**	**0.197**

† only the re-projected (sub)pixels

has the lowest scale-invariant inverse depth error in comparison with the relative depth prediction CNNs[10] (MiDaS and DiverseDepth) (see Figure 5.7 for a qualitative comparison). Due to the similarity between `Lab1` and `Lab2`, the depth prediction errors of MiDaS, DiverseDepth and pre-trained Monodepth2 are similar in both sequences. Conversely, our method can further boost the depth prediction accuracy and SVO map point accuracy (see the last and 1st row of Table 5.6) resulting from our proposed online adaptation. especially on a longer sequence (`Lab2`).

5.4.6 Runtime evaluation

The two main time-consuming processes in our proposed online adaptation framework are depth prediction and fine-tuning. On average, depth prediction and one fine-tuning iteration take about 18 ms and 250 ms to process, respectively. Incidentally, the global photometric BA time consumption varies according to the total number of photometric re-projections between the map points and keyframes, but real-time processing should not be affected as the BA process itself runs on another thread.

5.5 Summary

The key takeaways of this chapter are as follows.

[10]For relative depth prediction to achieve maximum accuracy, accurate scale- and shift-correction for each predicted depth map is required [83], [136]

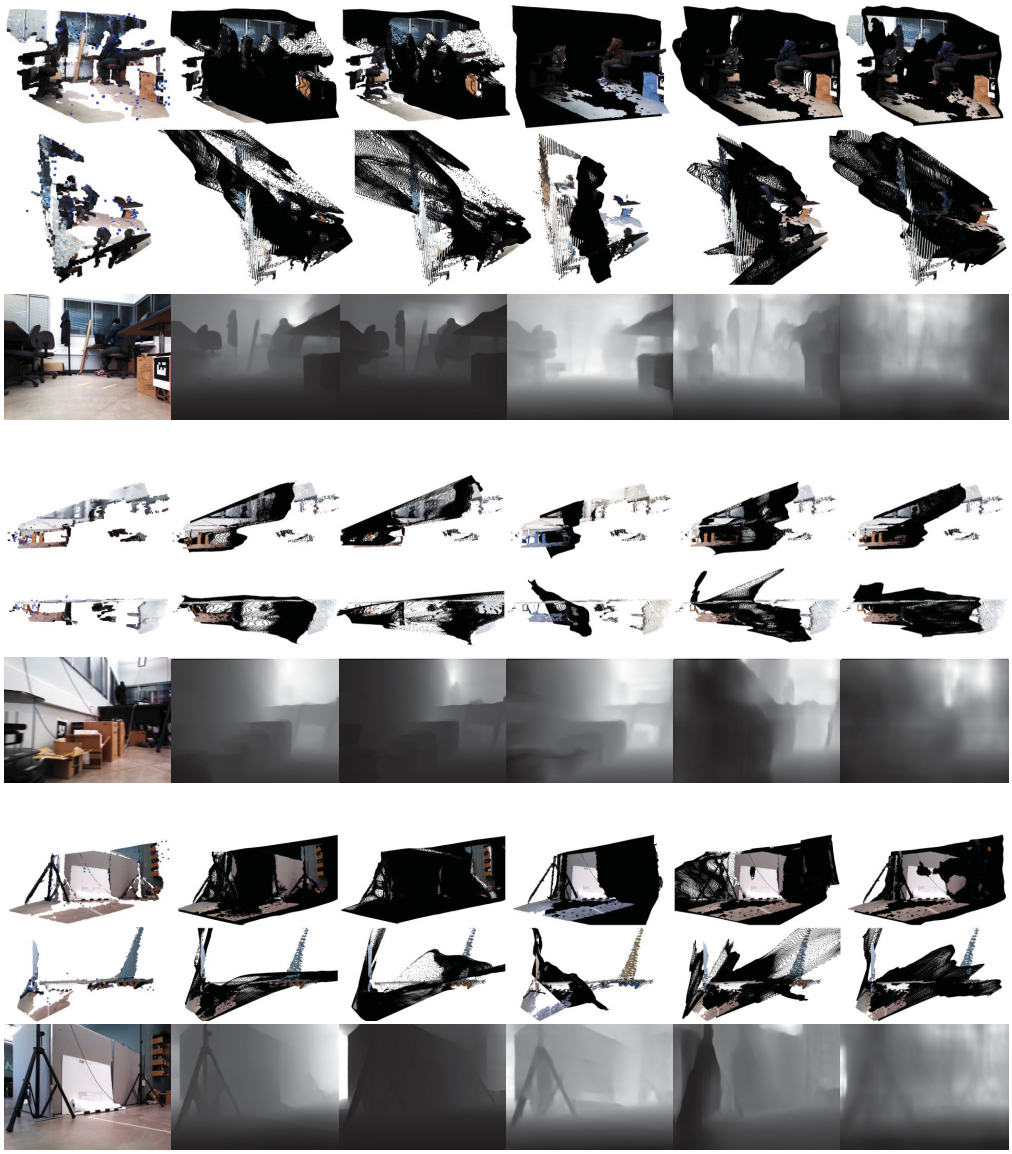

Figure 5.7: A qualitative comparison of the back-projected point clouds (shown in black) between (from left to right) ground truth depth with SVO map points (in blue), MiDaS `v2.1`, MiDaS `v3.0`, DiverseDepth, pre-trained Monodepth2, and our online adapted Monodepth2. From top to bottom: first and second viewpoint of the back-projected depth maps and the predicted depth maps by the aforementioned CNN models. Our proposed method's overall online adapted depth prediction accuracy compares favourably with MiDaS and DiverseDepth, which have been trained on an extensive collection of datasets. The predicted depth maps are scaled to ground truth. Best viewed digitally.

- We propose a novel online adaptation framework consisting of two complementary processes: a SLAM algorithm that is used to generate keyframes to fine-tune the depth prediction and another algorithm that uses the online adapted depth to improve map quality.

- Once the potential noisy map points are removed, we perform global photometric bundle adjustment (BA) to improve the overall SLAM performance.

- To perform online adaptation without *catastrophic forgetting*, we investigate the effectiveness of EWC regularization in the online adaptation of a depth prediction CNN, a single-task regression problem, as opposed to heavily studied regularization in multi-task classification problems [137], [138].

- Experimental results on both benchmark datasets and a real robot in our own experimental environments show that our proposed method improves the SLAM accuracy and that our online adapted depth prediction CNN outperforms the state-of-the-art depth prediction CNNs that have been trained on a large collection of datasets.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have demonstrated the effectiveness of incorporating single-image depth prediction to solve sparse and dense visual SLAM problems. Through extensive experiments on datasets and our own experimental environment, the experimental results have indicated that our proposed contributions excel in the following ways.

- In Chapter 3, we have improved the mapping of SVO [50], the state-of-the-art semi-direct SLAM, by initializing the map points with low depth uncertainty centred around the predicted depth from a single-depth prediction CNN. The proposed method has two main advantages. First, features can be matched effectively by limiting the search range along their respective epipolar lines, assuming known camera poses. Second, because the map points are initialized with low depth uncertainty, they can converge to their true depth values effectively. The experimental results have revealed improved camera tracking accuracy and robustness with the improved mapping performance. For the benefit of the robotics community, we have open-sourced this work, which is available at `https://github.com/yan99033/CNN-SVO`.

- In Chapter 4, we have presented a real-time dense monocular SLAM system that exploits the depth and depth gradient priors provided by single-image relative depth prediction. Our system densifies a semi-

dense structure provided by LSD-SLAM through a GPU-based energy minimization framework, of which the effectiveness of the error terms in the cost function has been examined through an ablation study. Our proposed adaptive filtering has been shown to remove spurious depth pixels in the semi-dense depth maps while preserving the structure, which in turn improves the densification accuracy. To improve the dense reconstruction accuracy further, we have presented a method that uses optimized depth maps to refine the keyframe poses. With accurate relative depth prediction on diverse scene types, the use of a relative depth prediction CNN is a promising step towards dense scene reconstruction in unconstrainted environments.

- In Chapter 5, we have addressed a practical robotics problem concerning the use of single-image depth prediction to improve SLAM performance. Notably, we have proposed a novel online adaptation framework in which the fine-tuning is enhanced with regularization for retaining the previously learned knowledge while the CNN is being trained continually. Also, we have demonstrated the use of fine-tuned depth prediction for map point culling before running global photometric BA, resulting in improved camera tracking and map reconstruction. Lastly, we have compared our online adaptation framework against state-of-the-art depth prediction CNNs that have been trained on a large number of datasets across domains, showing that our online adapted depth prediction CNN has a lower depth error, especially after performing online adaptation on a long sequence.

6.2 Limitations and Future Work

As a concluding remark, we would like to shed some light on our work's shortcomings and potential future research directions.

- Semi-direct and direct formulations in SLAM (e.g., SVO [50] and LSD-SLAM [37]) are promising in a way that algorithms work directly on raw

image pixels instead of having another layer of feature description computation in the algorithms. Although we have shown that single-image depth prediction is able to facilitate feature matching and regularization in the structure densification, the fundamental sparse and dense mapping problems could be improved. One way is to reformulate the sparse and dense mapping problems as the optical flow problem, i.e., considering the whole image when performing correspondences between image pixels [75], [139]. Alternatively, images can be projected to deep feature space using CNNs for facilitating the convergence to global optima when performing feature correspondence [140], [141]. After all, accurate and efficient feature matching is one of the cornerstones of reliable visual SLAM.

- We have been adopting modular approaches [130], [136], [142] to our sparse and dense SLAM system designs. While having additional modules in existing SLAM systems are effective and easy to debug (from a programmer's point of view), such system designs might not be as efficient and accurate as well-integrated systems, e.g., BA-Net [140] and DeepTAM [143]. The main difference between two design philosophies is that the latter leverages datasets to learn structure and motion optimizations by feedforwarding images through CNNs and backpropagating errors to iteratively optimizing camera poses and depth maps. In contrast, the former typically contains separate modules for solving dedicated tasks (e.g., camera tracking, graph optimization and loop-closing modules). Such fragmented designs can only be as strong as the weakest link, and our proposed contributions are designed to strengthen the weak links. A hybrid approach, such as ∇SLAM [144] and DROID-SLAM [145], which construct a fully differentiable computational graph in a traditional modular design of SLAM, could be a better solution for using deep learning to solve SLAM problems, alleviating the fragmentation issues in the modular design.

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[3] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[4] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," *arXiv preprint arXiv:2110.05352*, 2021.

[5] 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, China, May 2014.

[6] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, "Geonet: Geometric neural network for joint depth and surface normal estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 283–291.

[7] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[8] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[9] 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)*, IEEE, 2017, pp. 2524–2531.

[10] Z. Min, Y. Yang, and E. Dunn, "Voldor: Visual odometry from log-logistic dense optical flow residuals," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4898–4909.

[11] J. Tang, L. Ericson, J. Folkesson, and P. Jensfelt, "Gcnv2: Efficient correspondence prediction for real-time slam," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3505–3512, 2019.

[12] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proc. European Conference on Computer Vision (ECCV'18)*, Amsterdam, The Netherlands: Springer, Sep. 2018, pp. 817–833.

[13] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," pp. 1281–1292, 2020.

[14] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM-Learning a Compact, Optimisable Representation for Dense Visual SLAM," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, IEEE, Salt Lake City, Utah, Jun. 2018.

[15] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, IEEE, Honolulu, Hawaii, Jul. 2017.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3354–3361.

[17] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[18] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European conference on computer vision*, Springer, 2012, pp. 746–760.

[19] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[20] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, Springer, 2018, pp. 621–635.

[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[22] Nvidia. (). Omniverse isaac sim, [Online]. Available: `https://docs.omniverse.nvidia.com/app_isaacsim/app_isaacsim/overview.html` (visited on 11/19/2021).

[23] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE robotics & automation magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[24] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The slam problem: A survey," *Artificial Intelligence Research and Development*, pp. 363–371, 2008.

[25] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *IEEE Transactions on robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.

[26] 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*, vol. 29, no. 6, pp. 1052–1067, 2007.

[27] K. P. Murphy *et al.*, "Bayesian map learning in dynamic environments.," in *NIPS*, 1999, pp. 1015–1021.

[28] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, 2005, pp. 2432–2437.

[29] P. A. Gagniuc, *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[30] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[31] M. Kaess, A. Ranganathan, and F. Dellaert, "Isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[32] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[33] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[34] Y. Hou, H. Zhang, and S. Zhou, "Bocnf: Efficient image matching with bag of convnet features for scalable and robust visual place recognition," *Autonomous Robots*, vol. 42, no. 6, pp. 1169–1185, 2018.

[35] X. Zhang, L. Wang, and Y. Su, "Visual place recognition: A survey from deep learning perspective," *Pattern Recognition*, vol. 113, p. 107 760, 2021.

[36] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2015.

[37] J. Engel, T. Schps, and D. Cremers, "LSD-SLAM: Large-scale Direct Monocular SLAM," in *Proc. European Conference on Computer Vision (ECCV'14)*, Zurich, Switzerland: Springer, Sep. 2014, pp. 834–849.

[38] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2018.

[39] H. Lim, J. Lim, and H. J. Kim, "Real-Time 6-DOF Monocular Visual SLAM in a Large-Scale Environment," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'14)*, Hong Kong, China: IEEE, May 2014, pp. 1532–1539.

[40] S. Song, M. Chandraker, and C. C. Guest, "Parallel, Real-Time Monocular Visual Odometry," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany: IEEE, May 2013, pp. 4698–4705.

[41] Z. Dai, X. Huang, W. Chen, L. He, and H. Zhang, "A comparison of cnn-based and hand-crafted keypoint descriptors," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 2399–2404.

[42] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 716–12 725.

[43] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[44] R. Hartley and A. Zisserman, "Iterative estimation methods," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 597–627. DOI: 10.1017/CBO9780511811685.035.

[45] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny, "Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking," *Journal of Sensors*, vol. 2021, 2021.

[46] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Ieee, vol. 1, 2004, pp. I–I.

[47] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[48] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, IEEE, 2007, pp. 225–234.

[49] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*, IEEE, 2011, pp. 2320–2327.

[50] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'14)*, Hong Kong, China: IEEE, May 2014, pp. 15–22.

[51] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[52] X. Gao, R. Wang, N. Demmel, and D. Cremers, "Ldso: Direct sparse odometry with loop closure," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 2198–2204.

[53] M. Yokozuka, S. Oishi, S. Thompson, and A. Banno, "Vitamin-e: Visual tracking and mapping with extremely dense feature points," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9641–9650.

[54] S. Sumikura, M. Shibuya, and K. Sakurada, "Openvslam: A versatile visual slam framework," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2292–2295.

[55] R. Munoz-Salinas and R. Medina-Carnicer, "Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," *Pattern Recognition*, vol. 101, p. 107 193, 2020.

[56] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct sparse mapping," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1363–1370, 2020.

[57] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[58] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, 2021.

[59] Y.-M. Tsai, Y.-L. Chang, and L.-G. Chen, "Block-based vanishing line and vanishing point detection for 3d scene reconstruction," in *2006 international symposium on intelligent signal processing and communications*, IEEE, 2006, pp. 586–589.

[60] C. Tang, C. Hou, and Z. Song, "Depth recovery and refinement from a single image using defocus cues," *Journal of Modern Optics*, vol. 62, no. 6, pp. 441–448, 2015.

[61] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape-from-shading: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 8, pp. 690–706, 1999.

[62] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[63] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 179–12 188.

[64] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6647–6655.

[65] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, IEEE, Honolulu, Hawaii, Jul. 2017.

[66] W. Yin, Y. Liu, C. Shen, and Y. Yan, "Enforcing geometric constraints of virtual normal for depth prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5684–5693.

[67] 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*, Springer, 2016, pp. 740–756.

[68] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1851–1858.

[69] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1983–1992.

[70] 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*, 2017, pp. 5038–5047.

[71] X. Liu, A. Sinha, M. Unberath, M. Ishii, G. D. Hager, R. H. Taylor, and A. Reiter, "Self-supervised learning for dense depth estimation in monocular endoscopy," in *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, Springer, 2018, pp. 128–138.

[72] M. Klodt and A. Vedaldi, "Supervising the new with the old: Learning sfm from sfm," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 698–713.

[73] A. J. Amiri, S. Y. Loo, and H. Zhang, "Semi-supervised monocular depth estimation with left-right consistency using deep neural network," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2019, pp. 602–607.

[74] W. Yin, Y. Liu, and C. Shen, "Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[75] J. Kopf, X. Rong, and J.-B. Huang, "Robust consistent video depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1611–1621.

[76] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia, "Generative adversarial networks for unsupervised monocular depth prediction," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[77] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3d packing for self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2485–2494.

[78] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2002–2011.

[79] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4009–4018.

[80] D. Zoran, P. Isola, D. Krishnan, and W. T. Freeman, "Learning ordinal relationships for mid-level vision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 388–396.

[81] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," in *Advances in neural information processing systems*, 2016, pp. 730–738.

[82] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[83] W. Yin, X. Wang, C. Shen, Y. Liu, Z. Tian, S. Xu, C. Sun, and D. Renyin, "Diversedepth: Affine-invariant depth prediction using diverse data," *arXiv preprint arXiv:2002.00569*, 2020.

[84] A. Concha and J. Civera, "Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 5686–5693.

[85] A. Argiles, J. Civera, and L. Montesano, "Dense multi-planar scene estimation from a sparse set of images," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 4448–4454.

[86] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *2009 IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 80–87.

[87] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–8.

[88] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, *et al.*, "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[89] T. Laidlow, J. Czarnowski, and S. Leutenegger, "Deepfusion: Real-time dense 3d reconstruction for monocular slam using single-view depth and gradient predictions," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 4068–4074.

[90] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2930–2937.

[91] H. Alismail, B. Browning, and S. Lucey, "Photometric bundle adjustment for vision-based slam," in *Asian Conference on Computer Vision*, Springer, 2016, pp. 324–341.

[92] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[93] P. Sturm, S. Ramalingam, J.-P. Tardif, S. Gasparini, J. Barreto, *et al.*, "Camera models and fundamental concepts used in geometric computer vision," *Foundations and Trends® in Computer Graphics and Vision*, vol. 6, no. 1–2, pp. 1–183, 2011.

[94] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 141–148.

[95] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[96] B. C. Hall *et al.*, *Lie groups, Lie algebras, and representations: an elementary introduction*. Springer, 2003, vol. 10.

[97] J.-L. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization," *University of Malaga, Tech. Rep*, vol. 3, p. 6, 2010.

[98] H. Strasdat, "Local accuracy and global consistency for efficient visual slam," PhD thesis, Department of Computing, Imperial College London, 2012.

[99] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3607–3613.

[100] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*, Springer, 1999, pp. 298–372.

[101] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

[102] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[103] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[104] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *2010 IEEE computer society conference on computer vision and pattern recognition*, IEEE, 2010, pp. 2432–2439.

[105] G. Vogiatzis and C. Hernández, "Video-based, real-time multi-view stereo," *Image Vis. Comput.*, vol. 29, no. 7, pp. 434–441, 2011, ISSN: 0262-8856.

[106] W. N. Greene and N. Roy, "Metrically-scaled monocular slam using learned scale factors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 43–50.

[107] R. Wang, M. Schworer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3903–3911.

[108] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, pp. 1498–1505.

[109] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Joint Pattern Recognition Symposium*, Springer, 2010, pp. 11–20.

[110] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, IEEE, 1998, pp. 839–846.

[111] C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz, "Neural rgb (r) d sensing: Depth and uncertainty from a video camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 986–10 995.

[112] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

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

[114] J. T. Barron, "A general and adaptive robust loss function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4331–4339.

[115] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F.

d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[116] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NeurIPS Workshop*, 2017.

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

[118] 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)*, Oct. 2012.

[119] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*, IEEE, 2016, pp. 239–248.

[120] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," *arXiv preprint arXiv:1607.02555*, 2016.

[121] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[122] M. Knowles, V. Peretroukhin, W. N. Greene, and N. Roy, "Toward robust and efficient online adaptation for deep stereo depth estimation," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2021.

[123] Y. Kuznietsov, M. Proesmans, and L. Van Gool, "Comoda: Continuous monocular depth adaptation using past experiences," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2021, pp. 2907–2917.

[124] H. Luo, Y. Gao, Y. Wu, C. Liao, X. Yang, and K.-T. Cheng, "Real-time dense monocular slam with online adapted depth prediction network," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 470–483, 2018.

[125] Z. Zhang, S. Lathuilière, E. Ricci, N. Sebe, Y. Yan, and J. Yang, "Online depth learning against forgetting in monocular videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4494–4503.

[126] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[127] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[128] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3987–3995.

[129] D. Maltoni and V. Lomonaco, "Continuous learning in single-incremental-task scenarios," *Neural Networks*, vol. 116, pp. 56–73, 2019.

[130] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, "Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 5218–5223.

[131] N. Demmel, M. Gao, E. Laude, T. Wu, and D. Cremers, "Distributed photometric bundle adjustment," in *2020 International Conference on 3D Vision (3DV)*, IEEE, 2020, pp. 140–149.

[132] L. Prechelt, "Early stopping-but when?" In *Neural Networks: Tricks of the trade*, Springer, 1998, pp. 55–69.

[133] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.

[134] F. Kunstner, P. Hennig, and L. Balles, "Limitations of the empirical fisher approximation for natural gradient descent," *Advances in neural information processing systems*, vol. 32, 2019.

[135] L. Tiwari, P. Ji, Q.-H. Tran, B. Zhuang, S. Anand, and M. Chandraker, "Pseudo rgb-d for self-improving monocular slam and depth prediction," in *European Conference on Computer Vision*, Springer, 2020, pp. 437–455.

[136] S. Y. Loo, S. Mashohor, S. H. Tang, and H. Zhang, "Deeprelativefusion: Dense monocular slam using single-image relative depth prediction," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 6641–6648.

[137] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[138] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[139] T. Ke, T. Do, K. Vuong, K. Sartipi, and S. I. Roumeliotis, "Deep multiview depth estimation with predicted uncertainty," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 9235–9241.

[140] C. Tang and P. Tan, "Ba-net: Dense bundle adjustment networks," in *International Conference on Learning Representations*, 2018.

[141] X. Gu, W. Yuan, Z. Dai, C. Tang, S. Zhu, and P. Tan, "Dro: Deep recurrent optimizer for structure-from-motion," *arXiv preprint arXiv:2103.13201*, 2021.

[142] S. Y. Loo, M. Shakeri, S. H. Tang, S. Mashohor, and H. Zhang, *Online mutual adaptation of deep depth prediction and visual slam.*

[143] H. Zhou, B. Ummenhofer, and T. Brox, "Deeptam: Deep tracking and mapping," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 822–838.

[144] K. M. Jatavallabhula, G. Iyer, and L. Paull, "Gradslam: Dense slam meets automatic differentiation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 2130–2137.

[145] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[146] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *Journal of Field Robotics*, vol. 27, no. 5, pp. 587–608, 2010.

Appendix A

Online adaptation regularization

Regularization is a way to prevent catastrophic forgetting in the online adaptation of single-image depth prediction. Here, we present three regularization techniques: elastic weight consolidation (EWC) (Section A.1), synaptic intelligence (SI) and memory aware synapses (MAS) (Section A.2).

A.1 Adapted EWC regularization for single-task regression problem

To perform online adaptation incrementally on a single task, we measure and consolidate the importance of the CNN parameters at each iteration, allowing for the preservation of learned knowledge. The EWC loss comprises the training loss and a regularizer to fine-tune the CNN parameters while preserving the important parameters measured through consolidation.

Formally, let E be the loss function of the problem at hand and assuming E to be locally smooth. we may expand the function about θ_0 using Taylor series approximation:

$$E(\theta) = E(\theta_0) + \mathbf{J}_\theta(\theta_0)(\theta - \theta_0) + \frac{1}{2!}(\theta - \theta_0)^T \mathbf{H}_\theta(\theta_0)(\theta - \theta_0) + \text{h.o.t}, \tag{A.1}$$

where $\mathbf{J}_\theta(\theta_0)$ and $\mathbf{H}_\theta(\theta_0)$ are the Jacobian and Hessian of E evaluated at θ_0, respectively. Assume that the CNN is fine-tuned on *old* data $\mathcal{T}_1$, which implies that θ_0 would be at a local minimum (denoted by $\theta^*_{\mathcal{T}_1}$) and that $E'(\theta^*_{\mathcal{T}_1}) = 0$,

we can rewrite Equation A.1 as

$$E_{\mathcal{T}_1}(\theta) \approx E(\theta^*_{\mathcal{T}_1}) + \frac{1}{2!}(\theta - \theta^*_{\mathcal{T}_1})^T \mathbf{H}_\theta(\theta^*_{\mathcal{T}_1})(\theta - \theta^*_{\mathcal{T}_1}) \tag{A.2}$$

where $E_{\mathcal{T}_1}$ is the approximated error for *old* data $\mathcal{T}_1$. Next, to fine-tune the CNN on *recent* data $\mathcal{T}_2$, we can use the following loss function:

$$E(\theta) = E_{\mathcal{T}_2}(\theta) + \beta E_{\mathcal{T}_1}(\theta), \tag{A.3}$$

where β controls the weighting between the two loss terms. Substituting Equation A.2 into Equation A.3, we get

$$\begin{aligned} E(\theta) &= E_{\mathcal{T}_2}(\theta) + \beta\Big(E(\theta^*_{\mathcal{T}_1}) + \frac{1}{2}(\theta - \theta^*_{\mathcal{T}_1})^T \mathbf{H}_\theta(\theta^*_{\mathcal{T}_1})(\theta - \theta^*_{\mathcal{T}_1})\Big) \\ &= E_{\mathcal{T}_2}(\theta) + \frac{\beta}{2}\Big((\theta - \theta^*_{\mathcal{T}_1})^T \mathbf{H}_\theta(\theta^*_{\mathcal{T}_1})(\theta - \theta^*_{\mathcal{T}_1})\Big), \end{aligned} \tag{A.4}$$

in which the first term of Equation A.2 is being treated as a constant and can be eliminated in the optimization. To measure the importance of the parameters contained in the Hessian matrix, Kirkpatrick et al. approximate the Gaussian distribution of the posterior with Laplace approximation, whereby diagonal of the Fisher information matrix replaces the Hessian matrix for the approximation of the posterior [126]:

$$E(\theta) = E_{\mathcal{T}_2}(\theta) + \frac{\beta}{2}\Big(\sum_i \mathbf{F}_i(\theta_i - \theta^*_{\mathcal{T}_1,i})^2\Big). \tag{A.5}$$

A.2 SI and MAS regularizations

Similar to EWC regularization [126], SI and MAS can be expressed using the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{train}} + \lambda \sum_i \mathbf{\Omega}_i(\theta_i - \theta^*_i)^2, \tag{A.6}$$

with the main difference being the estimation of the weight importance matrix, $\mathbf{\Omega}_i$. For MAS, the weight importance matrix is given by the gradient of the squared L_2-norm of the CNN output function, G [127]:

$$\mathbf{\Omega}_i = \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \left\|G(x_j;\theta)\right\|_2^2}{\partial \theta_i}, \tag{A.7}$$

whereas for SI, the weight importance matrix is given by [128]:

$$\mathbf{\Omega}_i = \sum_j \frac{\omega_{ij}}{(\Delta\theta_{ij})^2 + \xi}, \tag{A.8}$$

where ω_{ij} is the estimated per-parameter contribution to the total loss, and $\Delta\theta_{ij}$ the total *trajectory* of the parameter.

Appendix B

Visual SLAM with factor graph optimization

Visual SLAM can be formulated as a factor graph optimization problem. In this thesis, we consider two types of factor graph optimization: photometric bundle adjustment (Section B.1) and pose-graph optimization (Section B.2). Let us consider a toy problem of visual SLAM (shown in Figure B.1) in which the solution (i.e., the optimal keyframe poses and map point positions) is constrained by the photometric re-projection factors and the odometry factors.

B.1 Photometric bundle adjustment

Photometric BA seeks to minimize the photometric re-projection errors defined by the photometric re-projection factors, as shown in Figure B.2. Formally, let the optimizable parameters be the keyframe poses $T = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$ and map points $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, where n and m are the number of keyframe poses and map points, respectively (according to Figure B.2, n and m would be 3 and 4, respectively). The maximum likelihood estimation of T and X can be solved by minimizing the least-squares errors from the photometric re-projection factors:

$$\underset{\{T_i\},\{X_j\}}{\arg\min} \frac{1}{2} \sum_{j=1}^{m} \sum_{i \in S(j)} \|\mathbf{e}_{ij}\|_2^2, \tag{B.1}$$

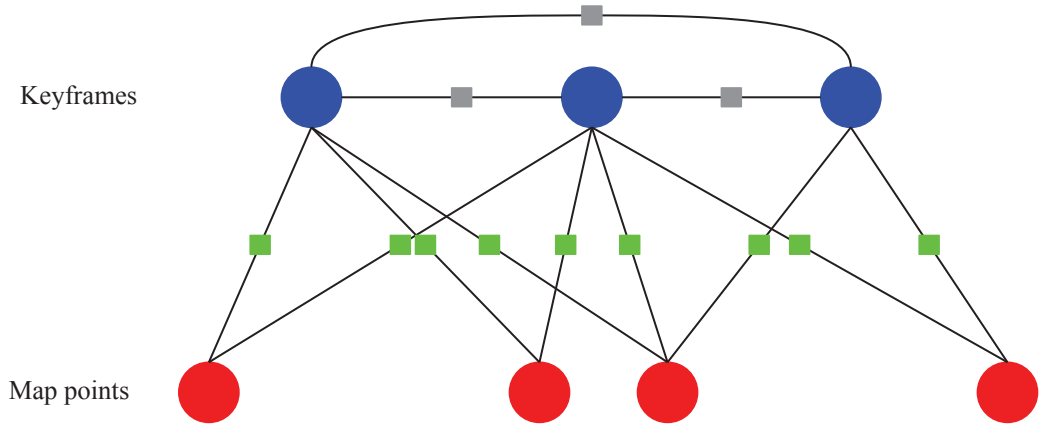

Figure B.1: Factor graph for optimizing the keyframe poses (blue nodes) and map points (red nodes) in visual SLAM. The green squares are the photometric re-projection factors, and grey squares the odometry factors.

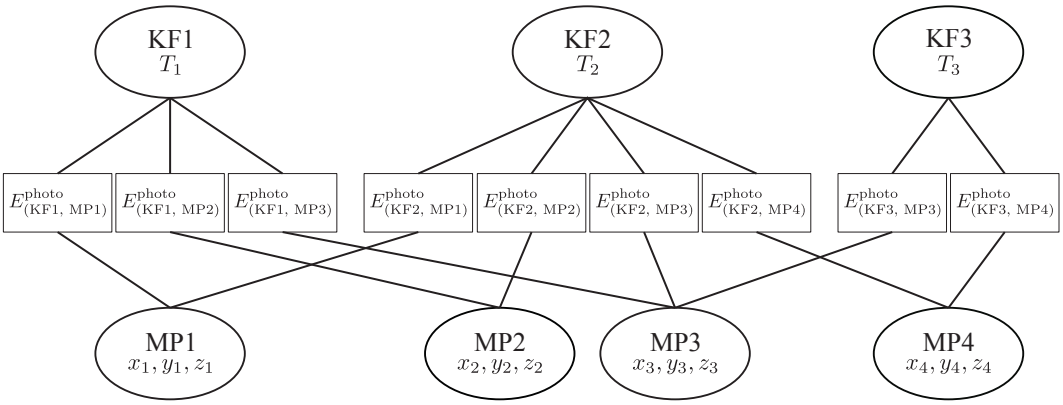

Figure B.2: Factor graph for solving photometric BA in which photometric re-projection factors ($E^{\text{photo}}_{(\cdot,\cdot)}$) are defined by projecting the map points (MPs) to their observable keyframes (KFs).

$S(\cdot)$ a set of keyframes observing $\mathbf{x}_j$, and $\mathbf{e}_{ij}$ the photometric residuals between the reference image patch and the projected image patch:

$$\mathbf{e}_{ij} = I_j(\mathbf{p}) - I_i(\mathbf{p}'), \tag{B.2}$$

with

$$\mathbf{p}' = \pi(\mathbf{R}_i\mathbf{x}_j + \mathbf{t}_i). \tag{B.3}$$

Further, $\mathbf{p}'$ is the re-projected image coordinates of $\mathbf{x}_j$ with the keyframe pose[1] $\mathbf{T}_i \in \mathrm{SE}(3)$, where

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}. \tag{B.4}$$

Iterative optimization of parameters

To iteratively optimize the map point locations as well as the keyframe poses over the SE(3) manifold, we make two following assumptions:

1. the total photometric error (in Equation B.1) has at least one well-defined minimum, and
2. the initial estimate of $\mathbf{u}$ (also known as $\mathbf{u}^{(0)}$) is close to a minimum.

Equation B.2 can be expressed more generally using the measurement function f and the corresponding variables $\mathbf{u}_{ij} = \begin{bmatrix}\mathbf{T}_i & \mathbf{x}_j\end{bmatrix}^T$ and reference image patch $\mathbf{y}_j$ for calculating the photometric residuals $\mathbf{e}_{ij}$:

$$\mathbf{e}_{ij} = \mathbf{y}_j - f(\mathbf{u}_{ij}). \tag{B.5}$$

Therefore, we may rewrite Equation B.1 as

$$\arg\min_{\mathbf{x}} \|\mathbf{y} - f(\mathbf{u}^{(0)})\|_2^2 = \arg\min_{\mathbf{x}} \|\mathbf{e}(\mathbf{u}^{(0)})\|_2^2, \tag{B.6}$$

where $\mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]^T$ the vector containing the reference image patch, $\mathbf{u} = [\mathbf{T}_1, \ldots, \mathbf{T}_n, \mathbf{x}_1, \ldots, \mathbf{x}_m]^T$ the vector containing the optimizable parameters, and $\mathbf{e}(\mathbf{u}^{(0)}) = \mathbf{e}^{(0)} = [\mathbf{e}_{11}, \ldots, \mathbf{e}_{nm}]^T$ the stacked photometric errors by the

[1]Here we use SE(3) camera pose as an example. However, the steps described for solving photometric BA should be able to generalize to Sim(3) camera pose.

initial estimation $\mathbf{u}^{(0)}$. We seek an incremental step $\mathbf{u}^{(1)} \leftarrow \mathbf{u}^{(0)} + \boldsymbol{\Delta}^{(0)}$ that minimizes the total photometric error, i.e,

$$\arg\min_{\mathbf{x}} \|\mathbf{e}(\mathbf{u}^{(1)})\|_2^2 \tag{B.7}$$

$$= \arg\min_{\boldsymbol{\Delta}} \|\mathbf{e}(\mathbf{u}^{(0)} + \boldsymbol{\Delta}^{(0)})\|_2^2 \tag{B.8}$$

$$\approx \arg\min_{\boldsymbol{\Delta}} \|\mathbf{e}(\mathbf{u}^{(0)}) + \mathbf{J}_{\mathbf{e}}\boldsymbol{\Delta}^{(0)}\|_2^2 \tag{B.9}$$

$$= \arg\min_{\boldsymbol{\Delta}} \|\mathbf{e}^{(0)} + \mathbf{J}_{\mathbf{e}}\boldsymbol{\Delta}^{(0)}\|_2^2 \tag{B.10}$$

$$= \arg\min_{\boldsymbol{\Delta}} \mathbf{e}^{(0)T}\mathbf{e}^{(0)} + 2\boldsymbol{\Delta}^{(0)T}\mathbf{J}_{\mathbf{e}}\mathbf{e}^{(0)} + \boldsymbol{\Delta}^{(0)T}\mathbf{J}_{\mathbf{e}}^T\mathbf{J}_{\mathbf{e}}\boldsymbol{\Delta}^{(0)}, \tag{B.11}$$

where $\mathbf{J}_{\mathbf{e}} = \frac{\partial \mathbf{e}}{\partial \mathbf{x}}$ is the Jacobian of the energy function $\mathbf{e}$ evaluated at $\mathbf{u}^{(0)}$. The increment vector $\boldsymbol{\Delta}^{(0)}$ can be obtained as follows

$$2\mathbf{J}_{\mathbf{e}}\mathbf{e}^{(0)} + 2\mathbf{J}_{\mathbf{e}}^T\mathbf{J}_{\mathbf{e}}\boldsymbol{\Delta}^{(0)} = 0 \tag{B.12}$$

$$\boldsymbol{\Delta}^{(0)} = (\underbrace{\mathbf{J}_{\mathbf{e}}^T\mathbf{J}_{\mathbf{e}}}_{=:\mathbf{H}})^{-1}\underbrace{(-\mathbf{J}_{\mathbf{e}}^T\mathbf{e}^{(0)})}_{=:\mathbf{b}}, \tag{B.13}$$

which is used to iteratively update the parameters:

$$\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \boldsymbol{\Delta}^{(i)} \tag{B.14}$$

until the solution has converged to a local minimum. Note that $\mathbf{H}$ is the Hessian matrix, the second derivative of $\mathbf{e}$, and using $\mathbf{J}_{\mathbf{e}}^T\mathbf{J}_{\mathbf{e}}$ to approximate the Hessian is also known as the Gauss-Newton method [44]. For the iterative update of keyframe poses, we adopt the left-multiplication convention, which is given by

$$\mathbf{T}^{(i+1)} = \boldsymbol{\Delta}_{\boldsymbol{\xi}}^{(i)} \boxplus \mathbf{T}^{(i)} \tag{B.15}$$

$$= \exp(\boldsymbol{\Delta}_{\boldsymbol{\xi}}^{(i)})\mathbf{T}^{(i)}, \tag{B.16}$$

where $\boxplus : \mathfrak{se}(3) \times \mathrm{SE}(3) \rightarrow \mathrm{SE}(3)$ is the left-multiplication operator for the camera parameter increment, and $\exp : \mathfrak{se}(3) \rightarrow \mathrm{SE}(3)$ is the exponential map.

Leveraging the sparsity of the Hessian matrix

Figure B.3 displays the underlying structure of the Jacobian and Hessian matrices. Instead of computing the inverse of the Hessian matrix in Equation B.13, we divide the Hessian matrix into sub-blocks to solve the system of

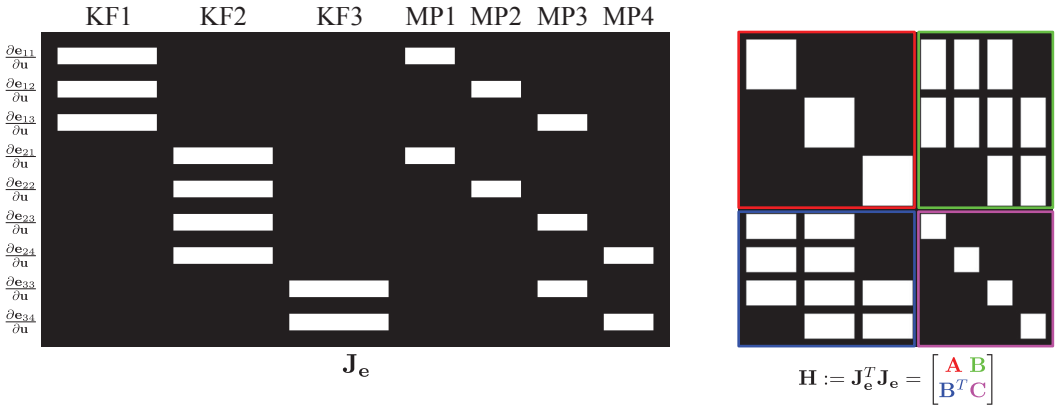

Figure B.3: Jacobian and Hessian matrices of the energy function. The matrices are fill with zeros except for the white blocks.

linear equations, i.e.,

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix}. \tag{B.17}$$

Thus, we can solve the system of linear equations $\mathbf{H\Delta} = \mathbf{b}$ (from Equation B.13) by performing a Gauss elimination on $\mathbf{H}$ as follows:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{\Delta_T} \\ \mathbf{\Delta_X} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{B.18}$$

$$\begin{bmatrix} \mathbf{I} & -\mathbf{BC}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{\Delta_T} \\ \mathbf{\Delta_X} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{BC}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{B.19}$$

$$\begin{bmatrix} \mathbf{A} - \mathbf{BC}^{-1}\mathbf{B}^T & \mathbf{0} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{\Delta_T} \\ \mathbf{\Delta_X} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 - \mathbf{BC}^{-1}\mathbf{b}_2 \\ \mathbf{b}_2 \end{bmatrix}. \tag{B.20}$$

We can first solve for the camera parameter increment ($\mathbf{\Delta_T}$)

$$(\mathbf{A} - \mathbf{BC}^{-1}\mathbf{B}^T)\mathbf{\Delta_T} = \mathbf{b}_1 - \mathbf{BC}^{-1}\mathbf{b}_2, \tag{B.21}$$

which is in turn used to solve for the map parameter increment ($\mathbf{\Delta_X}$). In practice, the number of map points is far greater than the number of keyframes ($m >> n$), hence a much larger submatrix $\mathbf{C}$. However, $\mathbf{C}$ is a block diagonal matrix, and its inverse can be carried out very efficiently. The partitioning of the Hessian matrix and leveraging the sparsity of the matrix is also known as the Schur complement trick for optimizing a large number of parameters in *sparse* SLAM problems [100], [146].

Weighted iterative estimation

Furthermore, one can increase the quality of the solution by weighting the photometric re-projection errors appropriately by introducing a positive-definite symmetric matrix $\mathbf{W_e}$ containing the weighting of the photometric re-projection factors to the system of linear equations:

$$\mathbf{J_e}^T\mathbf{W_e}\mathbf{J_e} = \mathbf{J_e}^T\mathbf{W_e}\mathbf{b}. \tag{B.22}$$

In practice, the weighting of the photometric re-projection factors can be calculated using a robust kernel function (e.g., Huber and Tukey norms).

B.2 Pose-graph optimization

On the other hand, pose-graph optimization seeks to minimize the keyframe pose errors defined by the odometry factors, as shown in Figure B.4. The maximum likelihood estimation of the keyframe poses $T = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$, where n is the number of keyframes, can be solved by minimizing the least squares errors from the odometry factors:

$$\arg\min_{\{T_i\}} \sum_{i=1}^{n} \sum_{j \in \varepsilon_i} \mathbf{e}_{ij}^T \mathbf{e}_{ij}, \tag{B.23}$$

where ε_i is a set of odometry constraints from the i-th keyframe and $\mathbf{e}_{ij}$ the odometry error between the two constrained keyframe poses:

$$\mathbf{e}_{ij} = \boldsymbol{\xi}_{ji} \circ \boldsymbol{\xi}_i^{-1} \circ \boldsymbol{\xi}_j \tag{B.24}$$

$$= \log(\mathbf{T}_{ij}^{-1}\mathbf{T}_i^{-1}\mathbf{T}_j) \tag{B.25}$$

where $\log : \mathrm{SE}(3) \to \mathfrak{se}(3)$ is the logarithmic mapping from a SE(3) keyframe pose ($\mathbf{T}_i, \mathbf{T}_j, \mathbf{T}_{ij} \in \mathrm{Sim}(3)$) to its corresponding Lie algebra pose ($\boldsymbol{\xi}_i, \boldsymbol{\xi}_j, \boldsymbol{\xi}_{ij} \in \mathfrak{se}(3)$). $\mathbf{T}_{ij}$ and its corresponding $\boldsymbol{\xi}_{ij}$ are an odometry constraint, and $\circ$ is the pose concatenation operator. The total pose-graph energy can be iteratively minimized using the Gauss-Newton or Levenberg-Marquardt method, similar to the steps elucidated in Section B.1.

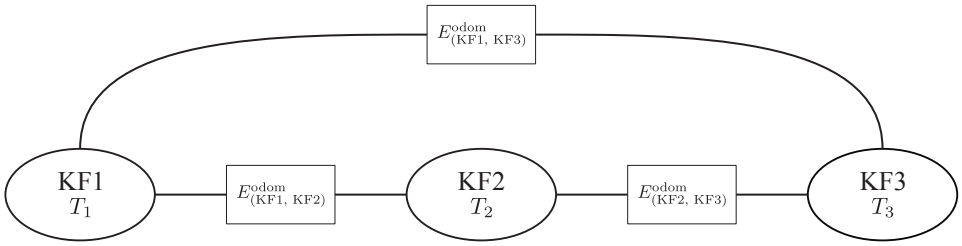

Figure B.4: Factor graph for optimizing the keyframe (KF) poses on the odometry constraints (from the odometry factors).

Weighted iterative estimation

Similar to photometric BA, odometry constraints can also be weighted according to their uncertainties:

$$\arg\min_{\{T_i\}} \sum_{i=1}^{n} \sum_{j \in \varepsilon_i} \mathbf{e}_{ij}^T \boldsymbol{\Sigma}_{ij}^{-1} \mathbf{e}_{ij}, \tag{B.26}$$

where the pose uncertainty $\boldsymbol{\Sigma}_{ij}$ may be obtained from the inverse of the Hessian $(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}$ from the last iteration of the camera tracking [37] or a pose uncertainty prediction CNN [13].