UNIVERSITY OF CALIFORNIA,
IRVINE

Geometric Reconstruction for Visual data Interpretation

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Minhaeng Lee

2019

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

As the first, I would like to acknowledge my advisor, Prof. Charless C. Fowlkes. From the beginning of my Ph.D program to the end, he has shown constantly generous attitude toward me with patient about my late progress and kind explanation even for my silly questions. His care and advising are the mainspring of my degree completion. It was great pleasure to be his student and he is the best advisor I have ever experienced.

Moreover, I would like to thank Prof. Ramesh Jain about his interest in my research topics, valuable discussion and fundamental questions. Also, I wish to thank Prof. Erik Sudderth about his effort to improve my thesis and writing through sharp feedbacks.

Also, I cannot miss my lab mates in computer vision lab: Bailey, James, Peiyun, Zhe, Daeyun, Samia, Yunhan, and Shu. We have spent more than 5 years together and built unforgettable memory in my life.

Finally, I would like to thank long period support from NFS IIS-1813785, IIS-1618806, and IIS-1253538.

CURRICULUM VITAE

Minhaeng Lee

EDUCATION

Doctor of Philosophy in Computer Science	**2019**
University of California,	*Irvine, California*
Master of Science in Computational Sciences	**2013**
Korea Advanced Institute of Science and Technology (KAIST)	*Daejeon, South Korea*
Bachelor of Science in Computational Sciences	**2008**
Soongsil University	*Seoul, South Korea*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2015–2019**
University of California, Irvine	*Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant	**2011–2012**
Korea Advanced Institute Science and Technology (KAIST)	*Daejeon, South Korea*

REFEREED CONFERENCE PUBLICATIONS

Minhaeng Lee and Charless Fowlkes, CeMNet: Self-supervised learning for accurate continuous ego-motion estimation **Jun 2019**
The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops

Minhaeng Lee and Charless Fowlkes, Space-Time Localization and Mapping **Sep 2017**
International Conference of Computer Vision (ICCV)

Ral Daz, Minhaeng Lee, Jochen Schubert, Charless Fowlkes, Lifting GIS Maps into Strong Geometric Context for Scene Understanding **Mar 2016**
Winter Conference On Applications of Computer Vision (WACV)

ABSTRACT OF THE DISSERTATION

Geometric Reconstruction for Visual data Interpretation

By

Minhaeng Lee

Doctor of Philosophy in Computer Science

University of California, Irvine, 2019

Charless C Fowlkes, Chair

Reconstruction happens in the human brain every day. When humans watch their surrounding scene, they effortlessly infer dynamic representations of scene geometry from sequences of images. This higher dimensional reconstruction not only helps to interpret the input data but also provides an important basis for performing complex, higher level tasks. In spite of the importance of scene reconstruction, it has been considered a difficult task since the amount of information in the input image (2D) is insufficient to fully reconstruct scene geometry (3D). Performing such a task clearly requires the use of prior knowledge. In this thesis, we explore the advantages of machine learning-based techniques in order to reconstruct geometric information from images or videos. We utilize deep neural networks and probabilistic models and demonstrate their effectiveness in reconstructing geometric information.

As the first part of this thesis, we estimate 2D motion flow from video, leveraging constraints of camera ego-motion and scene geometry. From a sequence of images, we first predict relative ego-motion between the input frames, and then reconstruct the camera trajectory. By considering the cycle consistency between 2D motion, depth and camera ego-motion, we train a model to reconstruct scene depth without additional supervision. These processes are trained using a self-supervised end-to-end convolutional neural network (CNN) architecture with motion field driven photometric consistency loss. To minimize accumulated error from imperfect local estimates, we predict relative reliability scores between every connected pair of input frames and then utilize them in global refinement.

In the second part, we reconstruct spatio-temporal 4D model from a set of 3D models. From a set of multiple 3D models, we optimize transformation parameters from each model space to global space using a Gaussian mixture to model point observations. We optimize alignment parameters using expectation-maximization algorithm and estimate the temporal extent for each 3D patches by maximizing expected posterior probability over time. This spatial-temporal model allows us to perform object segmentation as well as infer the existence of occluded objects.

Chapter 1

Introduction

1.1 Motivations

Inferring geometry from visual cues happens unconsciously in our real life and more frequent than we aware. This process helps us to understand our environment even from limited amount of information. The process includes reconstruction from lower dimension to higher, and sometimes we use our prior knowledge to fill out the gap between low and high dimensional data. In this thesis, we cover three aspects of inferring geometry : scene geometry, camera ego-motion, and temporal extent.

Scene geoemtry: Within a real environment, we recognize scene geometry or depth with stereopsis through our two eyes. However, we are able to infer geometry even with a single image. From a monocular image, since the amount of information is limited to reconstruct scene geometry, we use our prior knowledge to fill out the missing information. For example, we can use the approximated size of the objects we have seen before in order to reconstruct rough scene geometry. We can specify this inference process as the reconstruction from 2D to 2.5D which includes relative depth prediction.

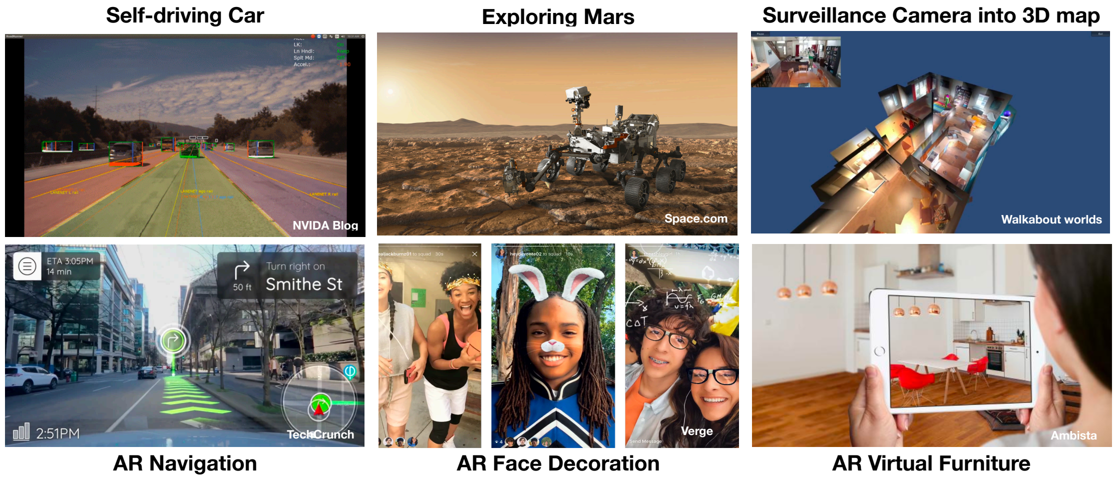


Figure 1.1: Example applications. If we know about scene geometry, then we can do many things.

Camera ego-motion: From motions in an image sequence, we can recognize the velocity and direction of objects relative to the camera. For example, the magnitude of motions when we walk and drive have big differences, and the magnitude implies how fast we were moving. Also, the angle of motions tells us how objects were moving relative to us. We call the 2D motion in the image optical flow and inferring the 2D motion helps us to estimate 3D camera motion.

Temporal extent: When we try to understand temporal scene geometry, we infer temporal interval of scene objects. An easy scenario is that after we lost our belongings and try to trace them. We might ask to ourselves a question ”when did I last see it?”. By answering this question, we can infer the temporal interval of objects. We also assume object permanence to reason about existence of objects even when they cannot be perceived. We can specify this process as a reconstruction from 2D video to 4D temporal extent.

If we train machines to do make these inferences, then we can build many useful applications as displayed in Figure 1.1 such as autonomous vehicle, exploring robot, surveillance camera system or augmented reality (AR) included navigation, face decoration and even virtual furniture placement[1].

[1]We do not focus on recognition part so every object in our output is considered as an ”object” but a specific object.

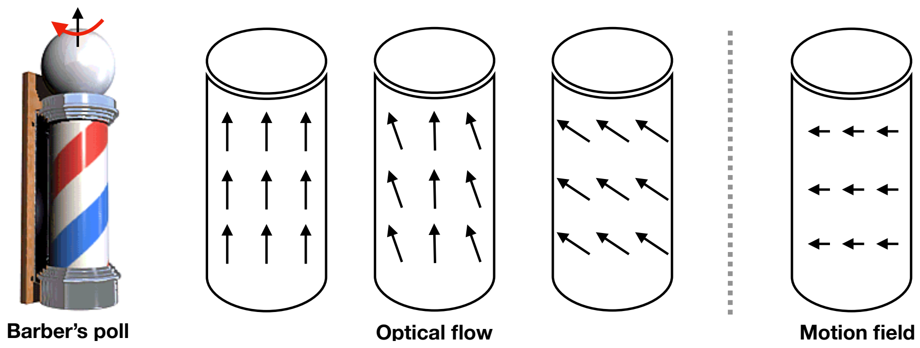


Figure 1.2: Illustration of difference between optical flow and motion field. While optical flow depends on visual appearance, motion field describes actual object motion.

1.2 Background

In this section, we briefly describe necessary background to understand how to get two essential things: camera poses and depth. We use the term camera ego-motion to mean the 6 degree of freedom (DoF) 3D motion including rotation and translation of an agent within an environment. For example, when a person is holding a camera and recording a scene while walking, then the person's motion is the ego-motion. We assume the scene geometry is static when we find camera pose in order to simplify the problem.

1.2.1 Optical flow and motion field

In this thesis, we use two different types of 2D motions: *optical flow* and *motion field*. Optical flow and motion field share common things but they are fundamentally different in terms of definition, applications, and methods to find them.

Definitions The optical flow describes a pattern of apparent motion of objects or surfaces in a visual scene caused by relative motion. The relative motion can caused by either the agent or the

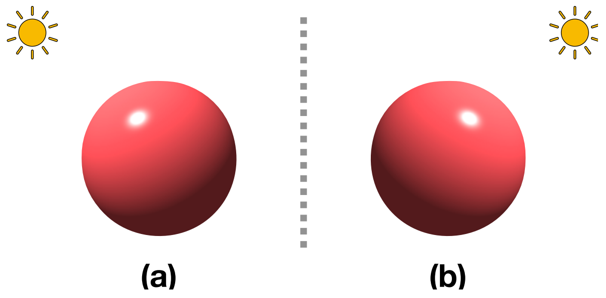


Figure 1.3: A sphere with different illumination sources. In this example, optical flow and motion field between (a) and (b) are completely different.

scene objects. Under the brightness consistency assumption, we can define the optical flow from given two images I_i and I_j. When a pixel coordinate x in image I_i and $x + \Delta x$ in I_j have similar pixel intensity, then we can call the 2D motion $(\Delta x, \Delta y)$ is the optical flow. We can describe the relationship between them as:

$$I_i(x, y) = I_j(x + \Delta x, y + \Delta y)). \tag{1.1}$$

In contrast, motion field is a representation of 3D motion projected into image space.

In a certain condition, the motion field looks similar to optical flow in terms of describing 2D motion. However, it represents physical 3D motions of objects while optical flow represents image appearance (*i.e.* pixel intensity). As displayed in Figure 1.2, we can see a fundamental difference between them. On the left, a barber's pole is rotating right to left. The rotating motion makes a visual illusion, the stripes look like they are moving up. So, as shown in the middle, we can find the corresponding optical flows. There are many different optical flows displayed at the middle since it only requires appearance constraint as described in Equation (1.1). However, in terms of motion field, the pole surface is moving right to left direction not bottom to up as visually seen. So there is only one motion field which can be drawn left directional arrows as shown in right image.

Another example displayed in Figure 1.3. There is a sphere with different illumination sources: left (a), and right (b). For both cases, the sphere is not moving but illumination source has moves. In this case, the motion field from (a) to (b) is completely zero but optical flow is not.

Applications Video compression and motion estimation are two popular applications using optical flow research. For the video compression, the estimated optical flow is used to reconstruct from previous frame to next frame so that it helps to compress the size of input video. Specifically, an image frame I_t can be reconstructed from I_{t-1} and estimated optical flow from $t-1$ to t-th frame $v_{t-1\rightarrow t}$. This allows compression since the estimated flow can be less expensive than image.

Another example is motion estimation for object tracking. In this case, the optical flow is used to find an actual motion of objects for tracking. The estimated optical flow is used as a proxy of motion field.

Classic methods There are several classical methods to find optical flow. The first example is Lucas-Kansde method [71] which assumes that the displacement of the image content nearby frame is small approximately constant within a certain size of window. Its limitation is that it only estimates constant optical flow within a given window and cannot recover dense optical flow. In contrast, Horn-Schunck method [45] assumes that the preferred flow is smooth over the entire image and tries to minimize irregular motions in flows. This method allows us to predict dense optical flow even in texture-less region but we need to set the weight of smoothness term carefully, and it is sensitive to noise.

There has been a large amount of work on optical flow in the intervening 38 years since these algorithms were first proposed (e.g., [30, 115, 89]). The primary focus has been on developing better regularization and optimization methods (e.g. using MRFs or variational methods), along with more explicit models of occlusion.

Optical flow with trainable networks There are a few limitations when we use classic methods for optical flow estimation. At first, we need to set regularization parameters carefully depending on input image distribution. As described above, the Horn-Schunck method has a smoothness regularization term and it determines the smoothness of estimated optical flow. Depending on the input image and preferred optical flow type, we need to set different weight parameter for the smoothness term. However, it is almost impossible to find the best weight for every possible input image. Also, it is hard to handle under-constrained scenes with classical methods. For example, for a texture-less scene, it is difficult to estimate fully filled optical flow since the region without texture has many trivial solutions that satisfy the photometric constraints described in Equation (1.1).

If we use a trainable network for optical flow estimation, then we can handle the limitations described above. Since the network includes very large number of trainable parameters to predict optical flow, it learns the regularization parameters naturally. Moreover, it is also possible to handle under-constrained scenes by training with a large number of training examples. With those advantages, learning based motion prediction methods are becoming more popular in modern computer vision communities.

Visual odometry from ego-motion and motion field

Since depth, camera ego-motion and the motion field have close relationship between them, we can estimate camera ego-motion given the depth and the motion field. In Figure 1.4, we display two possible cases of 3D motions. The first case (a), an object is static and camera is moving. Another case (b) is that the camera is static but the object is moving. In image space (c), those two cases show exactly the same 2D motion $x \rightarrow x'$. As a result, it is impossible to distinguish whether the camera or object is moving. In order to make the problem simpler, we assume that the camera is moving and consider the moving objects as the noise and outliers.

The 2D velocity in image space can be computed from 3D velocity. We map 3D object coordinates

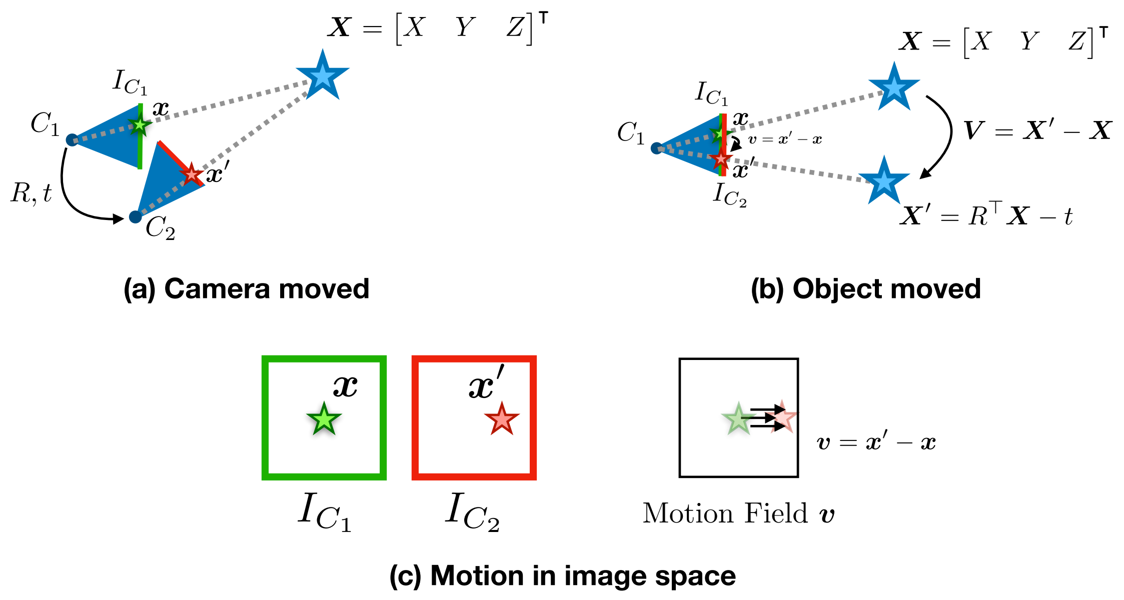

$\boldsymbol{X} = [X \quad Y \quad Z]^\intercal$
I_{C_1}
C_1
$\boldsymbol{x}$
$\boldsymbol{x}'$
R, t
C_2
(a) Camera moved
$\boldsymbol{X} = [X \quad Y \quad Z]^\intercal$
I_{C_1}
C_1
$\boldsymbol{x}$
$\boldsymbol{v} = \boldsymbol{x}' - \boldsymbol{x}$
$\boldsymbol{x}'$
I_{C_2}
$\boldsymbol{V} = \boldsymbol{X}' - \boldsymbol{X}$
$\boldsymbol{X}' = R^\top \boldsymbol{X} - t$
(b) Object moved
$\boldsymbol{x}$
$\boldsymbol{x}'$
I_{C_1}
I_{C_2}
$\boldsymbol{v} = \boldsymbol{x}' - \boldsymbol{x}$
Motion Field $\boldsymbol{v}$
(c) Motion in image space

Figure 1.4: Relationship between 3D motion and 2D motion. A camera C_1 watches an object X. (a)As the camera moves from C_1 to C_2, in camera viewpoint, the object is moved from x to x'. In the same way, when (b) the object moves from X to X' the 2D motion in image space is same as previous case. Thus, in image space, the 2D motion from x to x' cannot distinguish those cases since the motion is relative to each other. We simply assume that always moving part is camera while the objects are static.

into 2D according to the pinhole camera model. The 2D coordinate can be described using 3D coordinates as follows:

$$x, y = \frac{fX}{Z}, \frac{fY}{Z}, \tag{1.2}$$

where f mean focal length and Z mean depth of the 3D points. In order to get the movement at

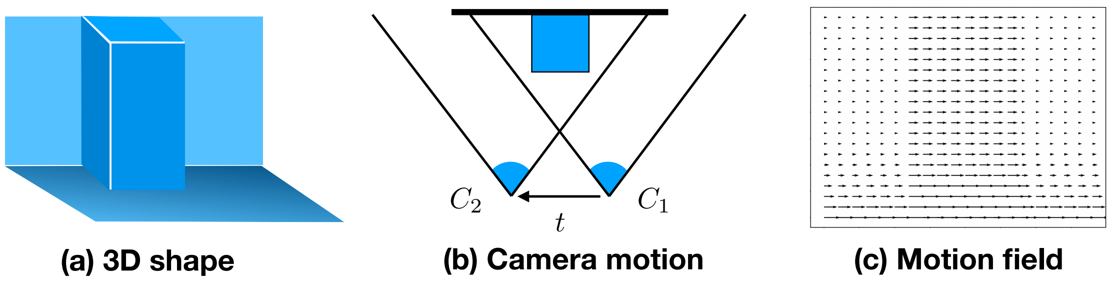


Figure 1.5: Relationship between 3D shape, camera motion, and corresponding motion field from camera viewpoint. (b) A camera moves from C_1 to C_2. In that case, the 2D motion in image space can be drawn as (c). When the distance from camera to object is closer to camera, then the motion becomes larger.

the point t, we take partial derivatives for both side with respect to time t.

$$\begin{aligned}\frac{\partial x(t)}{\partial t} &= \frac{\partial}{\partial t}\frac{fX(t)}{Z(t)} \\ &= \frac{f}{Z(t)}\frac{\partial X(t)}{\partial t} + fX(t)\frac{\partial}{\partial t}\frac{1}{Z(t)} \\ &= \frac{f}{Z(t)}\Delta X + fX(t)\Delta Z\end{aligned}$$

Now, we can convert 3D motion to 2D motion if we know the 3D point X and focal length of the camera. This mapping is used in Chapter 3 in order to estimate ego-motion from predicted 2D motions.

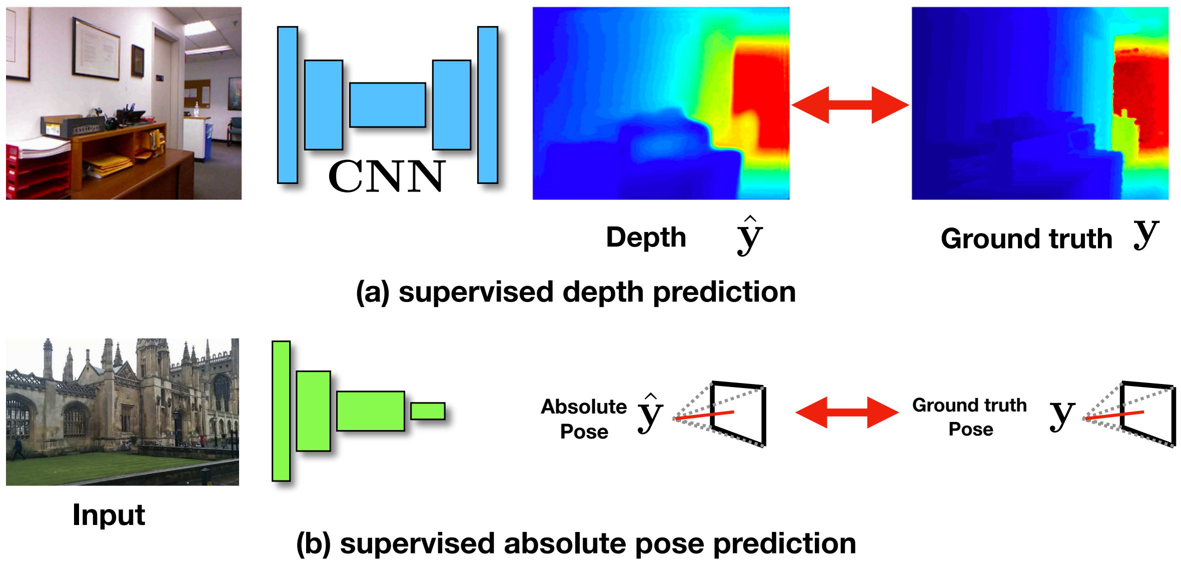


Figure 1.6: Example supervised learning for depth and pose.

1.2.2 Learning based methods

Supervised learning methods

With the development of stronger GPUs for faster numerical computation, deep learning methods opened a new era by solving many complicated tasks which had been considered difficult to solve such as single image pose localization, optical flow prediction, object recognition/classification, and natural language processing. As the fundamental building blocks, various novel network architectures had been introduced such as convolutional neural network (CNN), residual neural network (ResNet), or recurrent neural network (RNN). Especially in computer vision, CNN have made huge impact since CNN successfully extracts local features of image and allows faster training than fully connected layer (FC). When we have a good network architecture, then we can use supervised learning which is one of the simplest type of training. As shown in Figure 1.6, if we have a model f with a set of trainable parameter $\boldsymbol{\Theta}$, and the model maps an input data x to output data $\hat{y}$:

$$f_{\Theta} \;:\; x \to \hat{y}$$

During the training, a loss function $\mathcal{L}oss = d(\hat{y}, y)$ measures the distance between predicted label $\hat{y}$ and ground truth y. For example, Iro *et al.*[62] introduce simple depth prediction method from a single RGB image using fully convolutional residual network. Kendall *et al.*[55] train a camera localization method from a single RGB image (b). They simply feed the input RGB images and just regress 6 degree of freedom (DoF) camera poses.

If we have enough labelled data, then we can train almost any type of results with superior quality. However, there are several difficulties in collecting the data. First, those geometric related datasets are much more expensive than other purpose dataset such as object classification or detection. When we make a dataset for image classification, we just need to find the category index of given image and humans can do the labelling task. However, those geometric datasets which include absolute/relative depth or camera poses are almost impossible to generate using human annotations. Instead, we need to use sensor based devices such as depth cameras or motion sensors. Those devices are more expensive and less common than standard cameras.

Second, since the dataset is collected by sensor devices, its quality is also limited to the device specification such as the maximum length of range or sensitivity about noise. Moreover, each device has subtle physical differences between them, we need to do calibration steps to minimize the error from that. The calibration results also can have unexpected error.

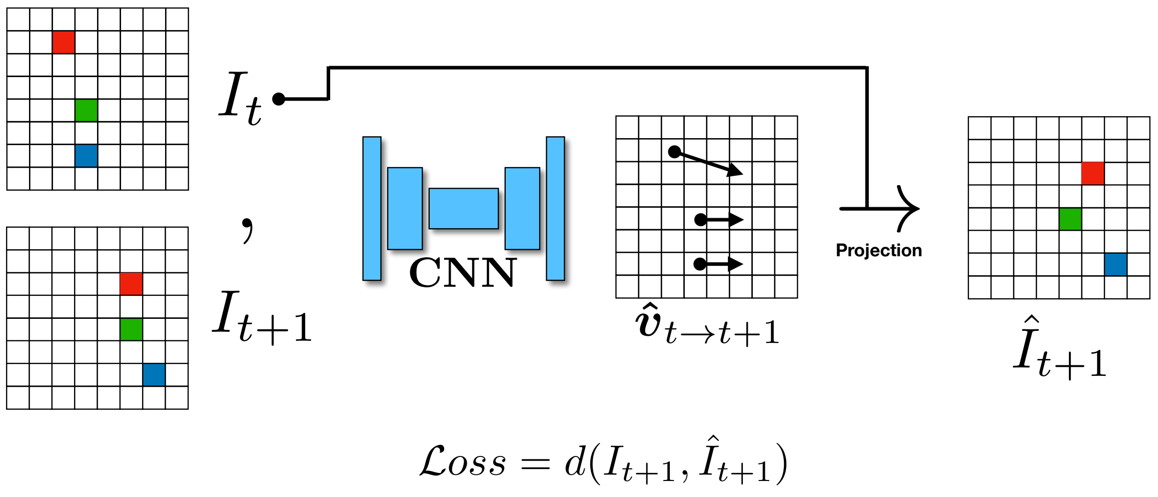

I_t
,
I_{t+1}
CNN
$\hat{\boldsymbol{v}}_{t\to t+1}$
Projection
$\hat{I}_{t+1}$
$\mathcal{L}oss = d(I_{t+1}, \hat{I}_{t+1})$

Figure 1.7: Example self-supervised learning for optical flow.

Self-supervised learning methods

In order to handle those difficulties of collecting labelled data, researchers started to develop unsupervised or self-supervised learning methods. As described above, the supervised training dataset consists of input x and preferred output y and those pair of dataset based training is the most straightforward type of training. Instead, with self-supervised or unsupervised learning, the preferred output y^* is extracted from input x by designing a label generating function L such as $L(x) \to y^*$.

Autoencoders are famous examples of learning method without label, and designed for representation learning. The architecture has bottleneck at the center of its network which enforces the compressed representation of the input. For the training, preferred output is input x itself. So the label function is $L(x) \to x$ and the network learns parameters by minimizing the loss function $\mathcal{L}oss = d(L(x), f(x))$.

Another example is optical flow learning. As described in Section 1.2.1, optical flow means apparent motion of objects. Specifically, if we have two images I_t and I_{t+1} and corresponding perfect 2D optical flow $(\Delta x, \Delta y)$, then pixel intensity at $I_t(x, y)$ and $I_{t+1}(x + \Delta x, y + \Delta y)$ should be same. From this constraint, we can define the label function $L(I_t, I_{t+1}) \rightarrow I_{t+1}$. In Figure 1.7, we displayed an example self-supervised learning for optical flow. From a pair of image (I_t, I_{t+1}), a trainable network (*e.g.* CNN) predicts optical flow $\hat{v}_{t \rightarrow t+1} = \{\hat{\Delta} x, \hat{\Delta} y\}$. Then we can train the model by minimizing the distance between target image I_{t+1} and projected image $\hat{I}_{t+1}$ from I_t with 2D motion $\hat{v}_{t \rightarrow t+1}$.

1.3 Contributions

The structure of this thesis is as follows: We first describe related works in Chapter 2. Then, in Chapter 3, we propose a novel self-supervised learning model for estimating continuous ego-motion from video. Our model learns to estimate camera motion by watching RGBD or RGB video streams and determining translational and rotation velocities that correctly predict the appearance of future frames. Our approach differs from other recent work on self-supervised structure-from-motion in its use of a continuous motion formulation and representation of rigid motion fields rather than direct prediction of camera parameters. To make estimation robust in dynamic environments with multiple moving objects, we introduce a simple two-component segmentation process that isolates the rigid background environment from dynamic scene elements. We demonstrate state-of-the-art accuracy of the self-trained model on several benchmark ego-motion datasets and highlight the ability of the model to provide superior rotational accuracy and handling of non-rigid scene motions.

As the extension of previous chapter, in Chapter 4, we present a self-supervised framework for predicting 6 DoF camera velocity from monocular RGB video which is refined by an efficient one-step update based on model gradients and subsequently integrated with global pose-graph optimization. We further investigate which local properties of the self-supervised reconstruction loss are predictive of the reliability of local pose estimation and utilize these features during global pose graph optimization. We evaluate the efficacy of these techniques on the car driving and drone racing datasets.

In Chapter 5, we addresses the problem of building a spatio-temporal model of the world from a stream of time-stamped data. Unlike traditional models for simultaneous localization and mapping (SLAM) and structure-from-motion (SfM) which focus on recovering a single rigid 3D model, we tackle the problem of mapping scenes in which dynamic components appear, move and disappear independently of each other over time. We introduce a simple generative probabilistic model of 4D structure which specifies location, spatial and temporal extent of rigid surface patches by local Gaussian mixtures. We fit this model to a time-stamped stream of input data using expectation-maximization to estimate the model structure parameters (mapping) and the alignment of the input data to the model (localization). By explicitly representing the temporal extent and observability of surfaces in a scene, our method yields superior localization and reconstruction relative to baselines that assume a static 3D scene. We carry out experiments on both synthetic RGB-D data streams as well as challenging real-world datasets, tracking scene dynamics in a human workspace over the course of several weeks.

Chapter 2

Related Works

Geometric reconstruction is a classic and well studied problem in computer vision area. Here we mention a few recent works that are most closely related to our approach.

Optical Flow, Depth and Odometry A number of recent papers have shown great success in estimation of optical flow from video using learning-based techniques [20, 48]. Ren *et al.* introduced unsupervised learning for optical flow prediction [90] using photometric consistency. Garg et. al. utilize consistency between stereo pairs to learn monocular depth estimation in a self-supervised manner [27]. [126] jointly trains estimators for monocular depth and relative pose using an unsupervised loss. SfM-Net [110] takes a similar approach but explicitly decomposes the input into multiple motion layers. [64] uses stereo video for joint training of depth and camera motion (sometimes referred to as scene flow) but tests on monocular sequences. Mahjourian *et al.*[73] use 3D ICP loss on top of 2D photometric loss to predict depth and ego-motion. Our approach differs from these recent papers in using a continuous formulation appropriate for video. Such a formulation was recently used by Jaegle *et al.*[51] for robust monocular ego-motion estimation but with classic (sparse) optical flow as input.

Self-supervised learning Making well-labeled dataset is always expensive and the performance of trained models are limited to the dataset quality. Moreover sometimes dataset can contain hidden biased or erroneous ground truth. However, with self-supervised training architecture, getting dataset is much cheaper and driven ground truth is much make sense than human labeled ground truth as long as the self-supervised architecture is well designed.

For those reason, the self-supervised learning getting popular by many vision applications. Doersch *et al.*[18] train contextual information by taking a pair of cropped image patches then train the relative position where they comes. The self-supervised training is also useful to training interest point and descriptor. Detone *et al.*[17] introduce joint training scheme to get trainable interest point detector and descriptor. Another example is optical flow learning in a sequence of video dataset. Meister *et al.*[77] introduce optical flow training scheme by utilizing forward/backward consensus loss with image warping. More recently, Liu *et al.*[70] proposed another optical flow learning by distilling reliable flow and followed by hallucinated occlusion.

Another popular application is depth training. From an input pair of images, their scene depth is closely related to relative camera pose. Clement *et al.*[33] choose simple reprojection loss, multi-scale sampling and auto-masking loss to avoid violated camera motion to achieve high quality depth prediction. Ariel *et al.*[37] use unsupervised depth estimation with predicted camera intrinsic matrix from YouTube random video dataset. Another depth and camera pose prediction from two-stream network with data augmentation is introduced by Ambrush *et al.*[5].

Learning based Visual Odometry Self-supervised learning is also popular for visual odometry system. One of the pioneer unsupervised VO is introduced by Zhou *et al.*[126]. They jointly train 6 DoF camera pose and depth from monocular video input with camera pose driven photometric consistency loss. With the similar unsupervised training scheme, Li *et al.*[65] proposed absolute scale recovery feature by adapting stereo dataset when they train their model. Yin *et al.*[123] introduce another self- supervised VO system by using forward/backward optical flow consistency.

Mahjourian *et al.*[74] use additional iterative closest point (ICP) based predicted depth aligning loss to get better pose and depth prediction.

Also, generative adversarial network (GAN) is used to synthesize realistic scene followed by reconstruction loss for better pose and depth training [4]. Meanwhile, the previous methods train relative pose from input RGB image pair directly, Lee and Fowlkes [63] use 2D motion prediction and then extract camera pose from the motion so that they divide pose prediction into two steps such as motion prediction and pose calculation. Also, Xue *et al.*[120] use recurrent neural network (RNN) to train the camera pose momentum and followed post refinement. Their method shows good performance especially in feature-less region.

SLAM While conventional simultaneous localization and mapping (SLAM) methods estimate geometric information by extracting feature points [56, 116] or use all information in the given images [21], recently several learning based methods have been introduced. Tateno *et al.* [103] propose a fusion SLAM technique by utilizing CNN based depth map prediction and monocular SLAM. Melekhov *et al.* propose CNN based relative pose estimation using end-to-end training with a spatial pyramid pooling (SPP) [78]. Other recent works [57, 66] model static background to predict accurate camera pose even in dynamic environment. Sun *et al.* try to solve dynamic scene problem by adding motion removal approach as a pre-processing to be integrated into RGBD SLAM [101]. The work of Wang *et al.* [112] train a recurrent CNN to capture longer-term processing of sequences typically handled by bundle adjustment and loop closure. [122] use virtual stereo camera based training to achieve photoconsistency and accurate depth reconstruction. Another recent work done by [7] shows scale-aware camera pose prediction by using spatial and temporal reconstruction losses simultaneously.

RGB-D SLAM Mapping from images has a long history in the robotics and computer vision literature with many recent developments motivated by emerging applications in 3D modeling, augmented reality and autonomous vehicles. Large-scale structure from motion (e.g., [2, 97, 23]) when combined with mult-view stereo (e.g., [24]) can yield rich geometric models but dense correspondence is often difficult to establish from monocular imagery, particularly for untextured surfaces common in indoor scenes.

The availability of cheap RGB-D sensors has enabled rapid progress indoor mapping, where active stereo or ToF provides very dense 3D structure and allows correspondence and pose estimation to be carried out by rigidly aligning scene fragments, e.g., using iterative closest point (ICP), rather than sparse matching and projective structure estimation techniques used in monocular SLAM. Initial work by Henry et al. [42] demonstrated the value of RGB-D data in ICP-based pose estimation while the KinectFusion system of Richard *et al.* [86] demonstrated impressive online reconstruction.

More recent work, such as ElasticFusion [116], has focused on improving performance of online real-time reconstruction and odometry by active updating and loop closure. To improve accuracy and robustness of offline reconstructions, Choi *et al.* used stronger priors on reconstructed geometry while carrying out global pose-graph optimization [14]. Recognition of familiar objects has also been integrated with SLAM-based approaches using prior knowledge of 3D structure [99] and fusing 2D semantic understanding with 3D reconstruction [43, 111]

Graph-based SLAM Many graph-based SLAM methods such as ORB-SLAM [82], LSD-SLAM [21] and the graph SLAM for urban structures [105] show successful result by building the connectivity of each frame as a graph. They mainly rely on 2D feature in order to find local camera motion as an edge of the graph, then apply global pose optimization such as bundle adjustment (BA).

Graph optimization is one of global optimization methods and well-studied conventional method to optimized absolute positions from given relative motions and information matrix [38]. Especially useful for closed loop correction [117]. Also, recently, Li *et al.*nsupervised ego-motion prediction followed by pose pose graph optimization [67]. Chen *et al.*[13] include on-line refinement stage in their training for predicting geometric consistency camera pose/parameters and depth prediction.

3D Registration A core component of contemporary SLAM approaches based on LiDAR or RGB-D sensors is estimating alignments between pointclouds from successive measurements. A traditional starting point is iterative closet point (ICP) [8] which refines a rigid alignment minimizing mean-square inter-point or point-to-surface distance. However, the RGB-D fragment alignment problem differs somewhat from the classic problem of aligning range scans (e.g., [3]) due to the narrow field of view which often lacks distinguishing geometric features.

Our approach is based on a family of methods that model geometry in terms of probability densities (rather than points, meshes or signed distance functions). Horaud *et al.* introduced expectation conditional maximization (ECM) method for rigid point registration [44]. This formulation is appealing as it avoids explicit point correspondences and naturally generalizes to multi-way registration [22] and non-rigid deformation models [83, 35]. Our model builds on the work of Evangelidis *et al.*, which uses an ECM-based formulation to align multiple point sets to a single underlying consensus model [22]. We augment this density model with a temporal dimension, occlusion reasoning, and a richer parameterization of local mixture components.

Dynamic Scenes and 4D maps Traditional SfM has primarily focused on recovering structure of a single rigid scene modeled by sparse keypoints. For dynamic scenes where correspondence is available in the form of extended keypoint tracks, multi-body SfM provides an approach to grouping tracks into subsets, each of which moves rigidly (e.g., [15, 121]) while non-rigid SfM (e.g., [9]) addresses recovery of non-rigid surfaces from such tracks. When correspondence is not available but smooth surfaces are densely sampled in space-time, surface tracking approaches can be used to fuse observations (e.g., [80, 85]). Here we focus on scenarios where the temporal sampling is too sparse to allow for effective surface or feature tracking.

Related work on the problem of geometry change detection from sparse imagery was investigated by [102] and [108], who detect geometric changes relative to an existing model using voxel-based appearance consistency to drive model updates. Change detection with viewability and occlusion was also explored by [34] for aligning observations to a construction job site plan. The work of [76] focuses on modeling dynamic appearance by grouping scene feature points into rectangular planes with an estimated temporal extent that captures, e.g., changing images on billboards. Finally, [75] used SfM to estimate geometry and warp images into a common viewpoint, enabling synthesis of time-lapse videos from unstructured photo collections. Closest in spirit to our approach is the work of Schindler and Dellaert [95] on “4D Cities”, which utilizes bottom-up heuristics for grouping point observations from an SfM pipeline into building hypotheses and a probabilistic temporal model to infer the time interval during which buildings exist.

Chapter 3

CeMNet: Self-supervised learning for accurate continuous ego-motion estimation

Supervised machine learning techniques based on deep neural networks have shown remarkable recent progress for image recognition and segmentation tasks. However, application of these powerful learning methods to geometric tasks such as structure-from-motion has been somewhat slower due to a number of factors. One challenge is that standard layers defined in convolutional neural network (CNN) architectures do not offer a natural way for researchers to incorporate hard-won insights about the algebraic structure of geometric vision problems, instead relying on general approximation properties of the network to re-discover these facts from training examples. This has resulted in some development of new building blocks (layers) specialized for geometric computations that can function inside standard gradient-based optimization frameworks (see e.g., [39, 47]) but interfacing these to image data is still a challenge.

A second difficulty is that optimizing convolutional neural networks (CNNs) requires large amounts of training data with ground-truth labels. Such ground-truth data is often not readily available for geometric problems (e.g., requiring special-purpose hardware during acquisition rather than simple image annotations). This challenge has driven recent effort to develop more realistic synthetic datasets such as Flying Chairs and MPI-Sintel [10] for flow and disparity estimation, Virtual KITTI [25] for object detection and tracking, semantic segmentation, flow and depth estimation, and SUNCG [98] for indoor room layout, depth and normal estimation.

In this chapter, we overcome some of these difficulties by taking a "self-supervised" approach to learning to estimate camera motions directly from video. Self-supervision utilizes unlabeled image data by constructing an encoder that transforms the image into an alternate representation and a decoder that maps back to the original image. This approach has been widely for low-level synthesis problems such as super-resolution [19], image colorization [124] and in-painting [88] where the encoder is fixed (creating a downsampled, grayscale or occluded version of the image) and the decoder is trained to reproduce the original image. For estimation tasks such as human pose [106], depth [110, 126], and intrinsic image decomposition [53], the structure of the decoder is typically specified by hand (e.g., synthesizing the next video frame in a sequence based on estimated optical flow and previous video frame) and the encoder is learned. Self-supervision is appealing for geometric estimation problems since (a) it doesn't require human supervision to generate target labels and hence can be trained on large, diverse data, and (b) the predictive (decoder) component of the model can incorporate known constraints into the problem structure.

Our basic model for ego-motion estimation takes a pair of calibrated RGB or RGBD video frames as input, estimates optical flow and depth, determines camera and object velocities, and resynthesizes the corresponding motion fields. We show that the model can be trained end-to-end with a self-supervised loss that enforces consistency of the predicted motion fields with the input frames, yielding a system that provides highly accurate estimates of camera ego-motion. We measure the effectiveness of our method using TUM [100] and Virtual KITTI [25] dataset.

Relative to other recent papers [107, 110, 126, 73, 7] that have also investigated self-supervision for structure-from-motion, the novel contributions of our work are:

- We represent camera motion implicitly in terms of motion fields and depth which are a better match for CNNs architectures that naturally operate in the image domain (rather than camera parameter space). We demonstrate that this choice yields better predictive performance, even when trained in the fully supervised setting

- Unlike previous self-supervised techniques, our model uses a continuous (linearized) approximation to camera motion [87, 51] which is suitable for video odometry and allows efficient backpropagation while providing strong constraints for learning from unsupervised data.

- Our experimental results demonstrate state-of-the-art performance on benchmark datasets which include non-rigid scene motion due to dynamic objects. Our model improves substantially on estimates of camera rotation, suggesting this approach can serve well as a drop-in replacement for local estimation in existing RGB(D) SLAM pipelines.

3.1 Continuous Ego-motion Network

Figure 3.1 provides an overview of three different types of architectures we consider in this chapter. We take as input a successive pair of RGB images $\{I_t, I_{t+\delta}\}$ and corresponding depth images $\{d_t, d_{t+\delta}\}$. When depth is not available, we assume it is predicted by a monocular depth estimator (not shown). The first network, $\boldsymbol{Net}_{\text{POSE}}$, directly predicts 6 DoF camera motion by attaching several fully connected layers at the end of a standard CNN architecture. When camera motion is known, this baseline can be trained with a supervised loss $\mathcal{L}_{\text{CAM}}$ or trained with a self-supervised image warping loss $\mathcal{L}_{\text{3DWARP}}$ as done in several recent papers [126, 107, 110].

Instead of directly predicting camera motion parameters, we advocate utilizing a fully-convolutional encoder/decoder architecture with skip connections (e.g., [93, 96, 20, 48]) to first predict optical flow (denoted $\boldsymbol{Net}_{\text{OF}}$). We then estimate continuous ego-motion (t, ω) using weighted least-squares and resynthesize the corresponding motion field $MF(t, \omega)$. These intermediate representations can be learned using unsupervised losses ($\mathcal{L}_{\text{OF}}$,$\mathcal{L}_{\text{MF}}$, $\mathcal{L}_{\text{OP}}$) described below. When additional moving objects are present in the scene, we introduce an additional segmentation network, $\boldsymbol{Net}_{\text{SEG}}$, which decomposes the optical flow into layers that are fit to separate motion models.

In the following sections we develop the continuous motion formulation, interpret our model as projecting the predicted optical flow on to the subspace of ego-motion flows, and discuss implementation of segmentation into layers.

3.1.1 Estimating Continuous Ego-motion

In this chapter, we use continuous formulation to describe camera ego-motion. As shown in Figure 3.4, we draw conceptual difference between a motion vector when we use Euler transformation and continuous formulation. Euler transformation can be approximated using continuous formulation if the angle Θ is small. We deal with video datasets and if the frame rate per second of them is large enough, then the approximation makes sense.

With continuous formulation, we can describe the relationship between 2D and 3D motions [41]. Consider the 2D trajectory of a point in the image $\boldsymbol{x} = \{x, y\}$ as a function of its 3D position $\boldsymbol{X} = \{X, Y, Z\}$ and motion relative to the camera. We write

$$\boldsymbol{x}(t) \;=\; \{x(t), y(t)\} = \left\{ \frac{fX(t)}{Z(t)}, \frac{fY(t)}{Z(t)} \right\},$$

where f is the camera focal length. To compute the projected velocity in the image $\boldsymbol{v}(\boldsymbol{x}) = (v_x, v_y)^\top \in \mathbb{R}^2$ as a function of the 3D velocity $\boldsymbol{V}(\boldsymbol{X})$ we take partial derivatives. For example, the x component of the velocity is:

$$\begin{aligned}\frac{\partial x(t)}{\partial t} &= \frac{f}{Z(t)}\frac{\partial X(t)}{\partial t} + fX(t)\cdot\frac{\partial}{\partial t}\frac{1}{Z(t)} \\ &= \frac{f}{Z(t)}V_x - fX(t)\cdot\frac{1}{Z^2(t)}\cdot\frac{\partial Z(t)}{\partial t} \\ &= \frac{1}{Z(t)}\begin{bmatrix} f & 0 & -x(t)\end{bmatrix}\begin{bmatrix} V_x \\ 0 \\ V_z\end{bmatrix}\end{aligned}$$

Dropping t for notational simplicity, we can thus write the image velocity as:

$$\boldsymbol{v}(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}A(\boldsymbol{x})\boldsymbol{V}(\boldsymbol{X}) \tag{3.1}$$

where the matrix $A(\boldsymbol{x})$ is given by:

$$A(\boldsymbol{x}) = \begin{bmatrix} f & 0 & -x \\ 0 & f & -y\end{bmatrix}.$$

In the continuous formulation, the velocity of the point relative to the camera $\boldsymbol{V}(\boldsymbol{X})$ arises from a combination of translational and rotational motions,

$$\boldsymbol{V}(\boldsymbol{X}) = \boldsymbol{\tau} + \boldsymbol{X}\times\boldsymbol{\omega}$$

where $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^\top \in \mathbb{R}^3$ is unit length axis representation of rotational velocity of the camera and $\boldsymbol{\tau} = (\tau_x, \tau_y, \tau_z)^\top \in \mathbb{R}^3$ is the translation. Denoting the inverse depth at image location

$\boldsymbol{x}$ by $\rho(x) = \frac{1}{Z(x)}$, we can see that the projected motion vector $\boldsymbol{v}$ is a linear function of the camera motion parameters:

$$\begin{aligned}
\boldsymbol{v}(\boldsymbol{x}) &= \rho(\boldsymbol{x})A(\boldsymbol{x})\boldsymbol{\tau} + B(\boldsymbol{x})\boldsymbol{\omega} \\
&= \begin{bmatrix} \rho(\boldsymbol{x})A(\boldsymbol{x}) & B(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau} \\ \boldsymbol{\omega} \end{bmatrix} \\
&= Q(\boldsymbol{x})\boldsymbol{T},
\end{aligned}$$

where the matrix B includes the cross product

$$B(\boldsymbol{x}) = \begin{bmatrix} -xy & f + x^2 & -y \\ -f - y^2 & xy & x \end{bmatrix}.$$

To describe motion field for the whole image, we concatenate equations for all N pixel locations and write $\mathcal{U} = \mathcal{Q}\boldsymbol{T}$ where

$$\begin{aligned}
\mathcal{U} &= \begin{bmatrix} \boldsymbol{v}(\boldsymbol{x}_1) \\ \boldsymbol{v}(\boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{v}(\boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{2N\times 1}, \\
\mathcal{Q} &= \begin{bmatrix} \rho_1 A(\boldsymbol{x}_1) & B(\boldsymbol{x}_1) \\ \rho_2 A(\boldsymbol{x}_2) & B(\boldsymbol{x}_2) \\ \vdots & \vdots \\ \rho_N A(\boldsymbol{x}_N) & B(\boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{2N\times 6}, \\
\boldsymbol{T} &\in \mathbb{R}^{6\times 1}.
\end{aligned}$$

We assume the focal length is a fixed quantity and in the following write the motion field as a function $\mathcal{U} = MF(\boldsymbol{\rho}, \boldsymbol{T})$ which is linear in both the inverse depths $\boldsymbol{\rho}$ and camera motion parameters $\boldsymbol{T}$.

To infer the camera motion $\boldsymbol{T}$ given inverse depths $\boldsymbol{\rho}$ and image velocities $\mathcal{U}$, we use a least-squares estimate:

$$\boldsymbol{t}^*, \boldsymbol{\omega}^* = \arg\min_{\boldsymbol{t},\boldsymbol{\omega}} \sum_{i=1}^{N} w(\boldsymbol{x_i}) || v(\boldsymbol{x}_i) - \frac{1}{Z(\boldsymbol{x}_i)} A(\boldsymbol{x}_i) t + B(\boldsymbol{x}_i) \omega ||^2$$

where $w(\boldsymbol{x_i})$ is a weighting function that models the reliability of each pixel velocity in estimating the camera motion. The solution to this problem can be expressed in closed from using the pseudo inverse of matrix $\mathcal{Q}$. We denote the mapping from $\mathcal{U}$ to estimated camera motion as $\boldsymbol{T} = MF^{\dagger}(\boldsymbol{\rho}, \mathcal{U}, \boldsymbol{w})$.

In our model we utilize $MF^{\dagger}(\boldsymbol{\rho}, \mathcal{U}, \boldsymbol{w})$ to estimate camera model and $MF(\boldsymbol{\rho}, \boldsymbol{T})$ to resynthesize the resulting motion field. Both functions are differentiable with respect to their inputs (in fact linear in $\mathcal{U}$ and $\boldsymbol{T}$ respectively) making it straightforward and efficient to incorporate them into a network that is trained end-to-end using gradient-based methods.

3.1.2 Projecting optical flow onto ego-motion

Given the true motion field $\mathcal{U}$, it is straight forward to estimate the the true camera motion $\boldsymbol{T}^*$. In practice, the motion must be estimated from image data which is often ambiguous (e.g., due to lack of texture) and noisy. Typically there is a large set of image flows that are photometrically consistent from which we must select the true motion field. Our architecture utilizes a CNN to generate an initial flow estimate from image data, then uses $MF^{\dagger}(\boldsymbol{\rho}, \mathcal{U}, \boldsymbol{w})$ to fit a camera motion and finally reconstructs the image motion field corresponding to the camera motion. The composition of $MF^\dagger$ and MF can be seen as a linear projection of the initial flow estimate into the space of continuous motion fields.

A key tenant of our approach is that it is a better match to convolutional feature extractors to predict the ego motion field in the image domain (and subsequently estimate camera motion) rather than attempting to directly regress camera pose. In particular, this allows for richer loss functions that guide the training of the network. We illustrate this idea schematically for the case of supervised learning in Figure 3.3. Panel (a) depicts the direct approach in terms of a loss function whose gradient pulls the predicted pose towards the true pose.

We display the relationship between optical flow, motion field and camera pose in Figure 3.3(b). Among all possible image flows ϕ, we indicate in yellow the set which are photometrically valid (i.e., have a zero warping loss $\mathcal{L}_{OF} \leq \epsilon$). The blue line indicates the 6-dimensional subspace consisting of those motion fields that can be generated by all possible camera velocities (conditioned on scene depth). Introducing a loss on the camera pose (either directly on the prediction τ, ω, or on the resynthesized motion field $MF(\tau, \omega)$ serves to pull the flow prediction towards the orthogonal complement of this space (i.e., the set $\{\phi : MF^\dagger(\phi) = \tau^*, \omega^*\}$ denoted by the gray vertical line).

Our approach allows the consideration of two other loss functions that can provide additional guidance. When supervision is available, we can utilize a loss which directly measures the distance between the predicted flow and the true motion field ($\mathcal{L}_*$ in the figure). In the self-supervised setting, we can approximate this with the photometric warping loss $\mathcal{L}_{OF}$. In either supervised or unsupervised settings, we can include an orthogonal projection loss $\mathcal{L}_{OP}$, which encourages the model to predict flows that are close to the space of motion fields. In section , we describe how these losses are computed and adapted to the unsupervised setting.

While all of these losses are minimized in a perfect model, Figure 3.3(c) shows that this choice of loss during training has a substantial practical effect. In the supervised setting, optimizing the direct loss in the camera pose space (using generic fully connected layers), or in the flow space (using our least-squares fitting) results in similar prediction errors. However, adding the projection loss or directly minimizing the distance to the true motion field yields substantially better predictions (i.e., halving average camera translation error).

3.1.3 Static and Dynamic Motion Layers

So far, our description has assumed a camera moving through a single rigid scene. A standard approach to modeling non-rigid scenes (e.g., due to relative motion of multiple dynamic objects in addition to ego-motion) is to split the scene into a number of layers where each layer has a separate motion model [114]. For example, Zhou *et al.* use a binary "explainable mask" [126] to exclude outlying motions, and Vijayanarasimhan *et al.* segment images into K regions based on motion [110]. However, in the later-case, there is no distinction between object motion and ego motion making it inappropriate for odometry.

We use a similar strategy in order to separate motion into two layers corresponding to static background and dynamic objects (outliers). We feed a pair of images and their predicted optical flow into a u-net-like segmentation network [93] (displayed in Figure 3.2) to predict this separation which then defines the weights used for camera motion estimation using pseudo inverse function $MF^{\dagger}(\cdot)$ described in Section 3.1.1.

Consider a scene divided into K regions corresponding to moving objects and rigid background. Let $Seg_i(x) \in \{0, 1\}$ denote a mask that indicates the image support of region i and $\mathcal{U}^i$ denote the corresponding rigid motion field for that object considered in isolation. The composite motion field for the whole image $\mathcal{U}$ can be written as:

$$\mathcal{U} = \sum_{i}^{K} Seg_i \cdot \mathcal{U}^i,$$

In the odometry setting, we are only interested in the motion of the camera relative to static background. We thus collect any dynamic objects into a single motion field and consider a single binary mask:

$$\mathcal{U}(\boldsymbol{x}_i) \approx Seg_s(\boldsymbol{x}_i)\mathcal{U}^s + Seg_d(\boldsymbol{x}_i)\mathcal{U}^d.$$

In our training with this segmentation network, we use the approximated motion field $\mathcal{U}$ for the photometric warping loss described below. For simplicity, we refer our single layer model as CeMNet^1 and dual layer model as CeMNet^2

In Figure 3.5, we illustrate intermediate results demonstrating how the 2 layer model can better estimate camera motion in the presence of dynamic objects. Since the single layer model cannot distinguish background and foreground, the quality of predicted camera pose is bad. Excluding the dynamic scene components from the camera motion estimation provides substantially better pose estimation as seen in panels (i) and (l) which show less photometric warping error on the scene background relative to the single layer model shown in (f).

Hard assignment to layers: Previous work such as [110] uses a soft probabilistic prediction of layer membership (i.e., using a softmax function to generate layer weights). However, such an approach introduces degeneracy since it can utilize weighted combinations of two motions to match the flow (e.g., even in a completely rigid scene). We find that using hard assignment of motions to layers yields superior camera motion estimates. We utilize the "Gumbel sampling trick" described in [109] to implement hard assignment while still allowing differentiable end-to-end training of both the flow and segment networks.

3.2 Training Losses

Losses for Self-supervision As described in Section 3.1.2, there are several different losses which can be applied to predicted flows. Here we adapt them to the self-supervised setting. The basic building block is to check if a predicted flow is photometrically consistent with the input image pairs.

For a given optical flow $\mathcal{U}^{OF}$ and source image $I_{t+\delta}$ then we can synthesize warped image $I_t^{\mathcal{W}_{OF}}$ and check if it matches I_t. As described in [50], this type of spatial transformation can be carried out in a differentiable framework using bilinear interpolation:

$$I_t^{\mathcal{W}^{OF}}(x_i) = \sum_{i \in \{t,b\}, j \in \{l,r\}} w^{ij} I_{t+\delta}(x_i + \mathcal{U}^{OF}(x_i)),$$

where w^{ij} denotes the bilinear weighting of the four sample points. For simplicity, we write $I_t^{\mathcal{W}^{OF}}(x_i) = \mathcal{W}(I_{t+\delta}, \mathcal{U}^{OF})$ to denote the warping of $I_{t+\delta}$ using flow $\mathcal{U}^{OF}$. We then define the self-supervised flow loss using the photometric error over all pixels:

$$\mathcal{L}_{OF} = \sum_{i=1}^{N} ||I_t(x_i) - I_t^{\mathcal{W}^{OF}}(x_i)||_1$$

This loss serves as an approximation of $\mathcal{L}_2$ when the predictions are far from the true motion field.

We can similarly apply warping loss is possible to the reconstructed motion field rather than the initial prediction. If the motion field we found is correct, then again, the warped image should be matched with the target image. We can build motion field loss by using motion-field warped image $I_t^{\mathcal{W}^{MF}} = \mathcal{W}(I_{t+\delta}, \mathcal{U}^{MF})$ as:

$$\mathcal{L}_{MF} = \sum_{i=1}^{N} P_t(x_i) ||I_t(x_i) - I_t^{\mathcal{W}^{MF}}(x_i)||_1$$

where the mask $P_t(x_i)$ is 1 when the depth at x_i is valid, 0 otherwise. This is necessary when using a depth sensor which doesn't provide depths at every image location. This loss acts as a proxy for minimizing the camera motion estimation error by lifting the prediction back to the flow space. When we predict camera motion for static scene, we use the global motion field, and for the dynamic scene, we use composite motion field $\mathcal{U}$.

Finally, we can utilize the orthogonal projection loss to minimize the distance between predicted optical flow and its projection onto the space of motion fields via:

$$\mathcal{L}_{OP} = \sum_{i}^{N} ||\mathcal{U}^{OF} - \mathcal{U}^{MF}||_1$$

By combining three above losses, we can define the final self-supervised loss function

$$\mathcal{L}_{Final} = \lambda_{OF}\mathcal{L}_{OF} + \lambda_{MF}\mathcal{L}_{MF} + \lambda_{OP}\mathcal{L}_{OP},$$

where λ_{OF}, λ_{MF} and λ_{OP} weigh relative importance (we use 1, 0.1 and 0.1 respectively in our experiments).

Semi-supervision for symmetry breaking In our segmentation network, we predict two layers corresponding to static and dynamic parts. However, in the unsupervised setting, the loss is symmetric with respect to which segment label is considered background. This symmetry problem can interfere with training of the model and affect final performance. To break this symmetry, we found it most effective to utilize a small amount of supervised data where camera motion is known. For the supervised data we use an additional loss term on the camera motion estimated for the background layer.

Our network predicts camera motion in an axis-angle representation that includes translation part $\boldsymbol{t} \in \mathbb{R}^3$ and rotation $\boldsymbol{w} \in \mathbb{R}^3$. For supervised loss, we treat these two components separately in order to match the criteria typically used in benchmarking pose estimation performance.

Following [100], we compute the difference between our predicted camera motion and the ground truth $\boldsymbol{Q}^d = (\boldsymbol{Q}^p)^{-1}\boldsymbol{Q}^{gt}$ where $\boldsymbol{Q} \in \mathbb{R}^{4\times4}$ and penalize the translation and rotation components

Seq.	DVO-SLAM [56]	Kintinuous [118]	ElasticFusion [116]	ORB2 [82]	CeMNet(RGBD)
fr1/desk	0.021	0.037	0.020	0.016	**0.0089**
fr1/desk2	0.046	0.071	0.048	0.022	**0.0129**
fr1/room	0.043	0.075	0.068	0.047	**0.0071**
fr2/xyz	0.018	0.029	0.011	0.004	**0.0009**
fr1/office	0.035	0.030	0.017	0.010	**0.0041**
fr1/nst	0.018	0.031	0.016	0.019	**0.0117**
fr1/360	0.092	-	-	-	**0.0088**
fr1/plant	0.025	-	-	-	**0.0061**
fr1/teddy	0.043	-	-	-	**0.0139**

Table 3.1: Relative translation error on TUM [100] static dataset. Most of the methods in this table use RGBD frames camera for pose prediction. Our model is trained without any supervised data.

respectively by:

$$\begin{aligned}\mathcal{L}_{trans} &= ||Q_t^d||_2 \\ \mathcal{L}_{rot} &= \arccos\left(\min\left(1, \max\left(-1, \frac{Tr(Q_r^d)-1}{2}\right)\right)\right)\end{aligned}$$

3.3 Experimental Results

For the following experiments, we use the synthetic Virtual KITTI dataset [25] depicting street scenes from a moving car, and the TUM RGBD dataset [100] which has been used to benchmark a variety of RGBD odometry algorithms. To measure performance, we use relative pose error protocol proposed in [100].

Self-supervised learning improves model performance: To show the benefits of self-supervision, we assume that only 10% of each dataset has ground-truth available. We use 11 different sequences from the TUM dataset as training, choose a random ordering of frame pairs over the whole dataset and train models with increasingly large subsets of the data and test on a separate held-out collection of frames. This allows us to evaluate the effect of growing the amount of supervised/unsupervised training data in a consistent way across models.

In Figure 3.6, we plot the relative translation/rotation errors as a function of training data size. The supervised version of the model (CeM-Sup) can only be trained on the first 10% of the dataset and makes no use of the unsupervised data. In this setting it outperforms the unsupervised model (CeM-Unsup). However, as the amount of unsupervised training data continues to grow, CeM-Unsup eventually outperforms the supervised model. For a clear comparison, the unsupervised losses are not used in training (CeM-Sup). We also compare a model which uses both supervised and unsupervised loss (CeM-SemiSup) which generally yields even better performance. We note that because the real world depth data in TUM is incomplete, limiting performance of the supervised model while the supervised model shows expected decreasing errors on Virtual KITTI.

Motion field and warping: In Section 3.2, we describe how a predicted camera pose is used to generate motion field and used in the warping loss. In Figure 3.7, we plot the per-pixel warping loss for several inputs. Left two (a-b) show the input RGB frames, (c) shows predicted optical flow. (d) is regenerated motion field. (e) shows differences between the target image and warped image. Note that blue color means lower differences between those two images.

Camera motion error comparison: To measure the quality of predicted camera pose, we compare our single layer model (CeMNet) with previous RGBD SLAM methods on the TUM dataset in Table 3.1. CeMNet(RGBD) shows the best average performance among tested methods in terms of relative translation error. Several previous methods of interest, including [126, 110] do not utilize depth as an input, instead predicting it directly from input images.

For fair comparison, we also test our model with predicted depth (CeMNet(RGB)) using off-the-

Seq.	TUM [100]		SfM-Net [110]		CeMNet(RGB)	
	Trans	Rot	Trans	Rot	Trans	Rot
fr1/desk	**0.008**	0.495	0.012	0.848	**0.0113**	**0.6315**
fr1/desk2	0.099	0.61	**0.012**	0.974	0.0133	**0.7548**
fr1/360	0.099	0.474	**0.009**	1.123	0.0091	**0.5455**
fr1/plant	0.016	1.053	0.011	0.796	**0.0083**	**0.5487**
fr1/teddy	0.020	1.14	0.0123	0.877	**0.0113**	**0.6460**

Table 3.2: To compare our model to RGB odometry methods, we use an off-the-shelf monocular depth estimator [62].

	Training		Testing		
	GT Depth	GT Cam	GT Depth	Trans	Rot
Geometric [51]	-	-		0.4579	0.3423
AIGN-SfM [107]	✓	✓		0.1247	0.3333
CeMNet(RGBD)	✓		✓	**0.0878**	**0.0781**
CeMNet(RGB)				0.0941	0.1079

Table 3.3: Relative pose error comparison using Virtual KITTI [25]. Both with (CeMNet(RGBD)) and without (CeMNet(RGB)) depth inputs, our models outperform previous methods.

shelf the monocular depth prediction model introduced by Iro *et al.* [62] which was trained using NYU Depth dataset V2 [84]. We rescale the predictions by 0.9 to match the range of depths in TUM (presumably due to differences in focal length) but otherwise leave the model fixed. As shown in Table 3.2, our method continues to outperform others in terms of rotation and shows comparable translation errors.

Additionally, we show performance on the Virtual KITTI dataset in Table 3.3. We specify how each method uses the available ground truth depth and camera pose data available for train and test. Using the true depth at test time results in strong performance from our model. For fair comparison, we also evaluate our model using the monocular depth prediction model of [31] trained

Seq.	Baseline		CeMNet[1]		CeMNet[2]		CeMNet[2](Semi)	
	Trans	Rot	Trans	Rot	Trans	Rot	Trans	Rot
fr3/sit_static	0.0134	0.5724	0.0025	0.1667	0.0016	0.1573	**0.0010**	**0.1527**
fr3/sit_xyz	0.0179	0.7484	0.0070	0.2645	0.0068	0.2653	**0.0064**	**0.2612**
fr3/sit_halfsph	0.0104	1.0135	0.0081	**0.5272**	0.0080	0.5820	**0.0074**	0.5552
fr3/walk_static	0.0149	0.5703	0.0103	0.2107	0.0030	0.1610	**0.0019**	**0.1583**
fr3/walk_xyz	0.0174	0.7952	0.0128	0.3338	0.0079	**0.2915**	**0.0078**	0.2921
fr3/walk_halfsph	0.0166	0.9426	0.0147	0.4698	0.0107	0.4120	**0.0102**	**0.3989**

Table 3.4: Relative pose error comparison using TUM dynamic dataset [100]. Generally, the two layered model shows better performance than single layered model. Including a small amount of supervision (CeMNet$^2(Semi)$) yields equivalent or better performance by breaking the symmetry of the unsupervised loss.

with KITTI [29] dataset and converted from the predicted disparity to depth[1]. The results show better performance than previous self-supervised approaches even without using ground-truth depth.

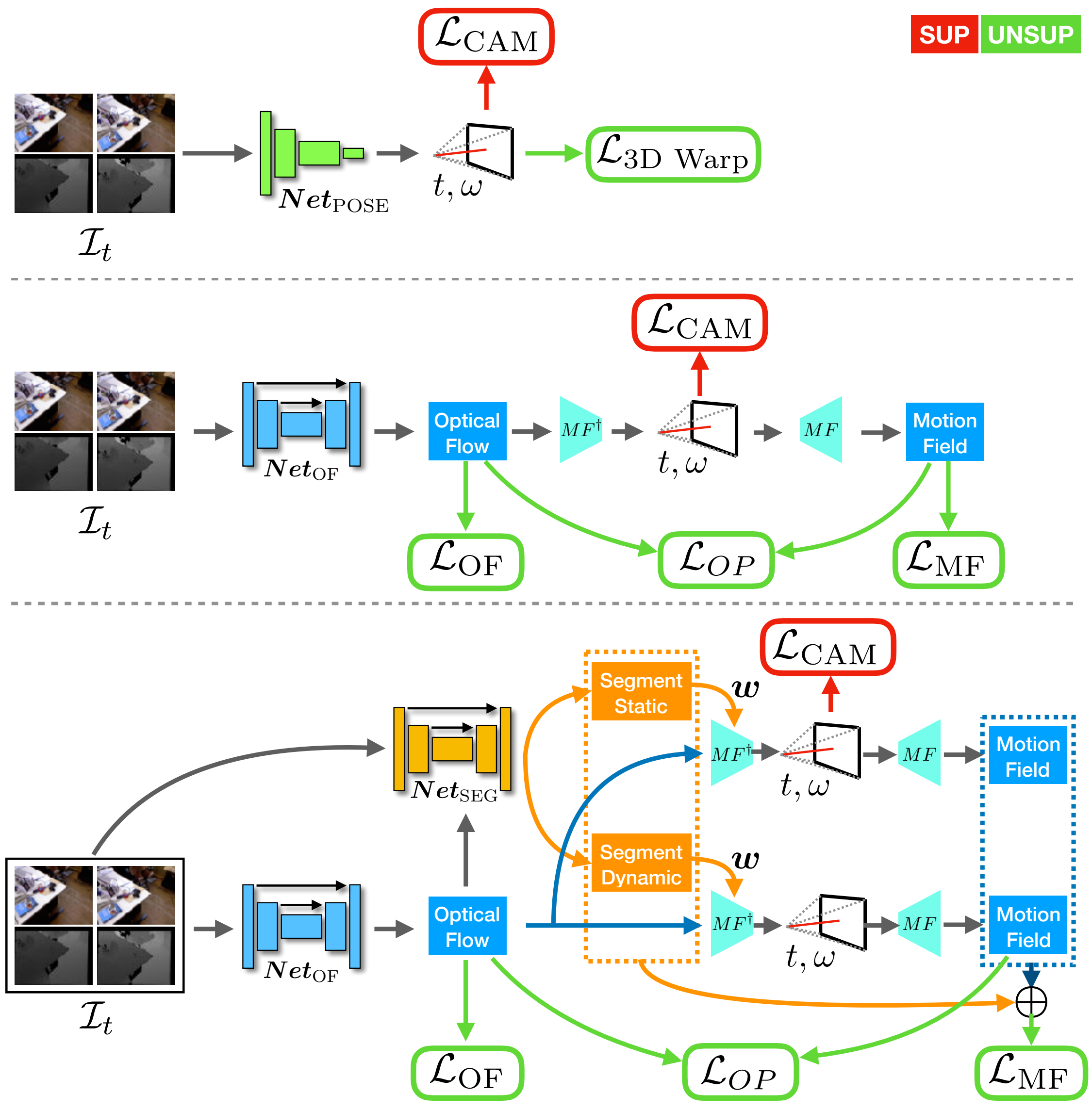
Static/Dynamic segmentation: In Figure 3.8, we visualize the results of breaking the input into static and dynamic layers. From the RGB input pair at I_t (a) and $I_{t+\delta}$, predicted optical flow is shown in (b). While single layered model generates motion field using the complete flow, the two layer model fits separate motions which segments moving objects and yields reduced warping error ((c) vs (f)), especially in the static background region.

We perform a quantitative comparison on the TUM dynamic dataset which includes both object and camera motion. The results results are shown in Table 3.4. While single layered models such as the baseline direct prediction model and CeMNet^1 are sensitive to dynamic objects, two layered model CeMNet^2 shows less pose error. However, as noted previously, the unsupervised loss suffers from a symmetry as to which layer correspond to ego-motion. We evaluate the use of a small amount of supervised data (10%) to break this symmetry in the segmentation prediction network. This yields the the lowest resulting motion errors across nearly all test sequences.

[1]We use 0.54 as baseline distance and 725 for focal length

3.4 Discussion

In this chapter, we have introduced a novel self-supervised approach for ego-motion prediction that leverages a continuous formulation of camera motion. This allows for linear projection of flows into the space of motion fields and (differentiable) end-to-end training. Compared to direct prediction of camera motion (both our own baseline implementation and previously reported performance), this approach yields more accurate two-frame estimates of camera motions for both RGBD and RGB odometry. Our model exploits self-supervised training, allowing it to make effective use of “free” unsupervised data. Finally, by utilizing a two-layer segmentation approach makes the model further robust to the presence of dynamic objects in a scene which otherwise interfere with accurate ego-motion estimation.



SUP
UNSUP
$\mathcal{L}_{\text{CAM}}$
$\mathcal{I}_t$
$\boldsymbol{Net}_{\text{POSE}}$
t, ω
$\mathcal{L}_{\text{3D Warp}}$
$\boldsymbol{Net}_{\text{OF}}$
Optical Flow
$MF^{\dagger}$
MF
Motion Field
$\mathcal{L}_{\text{OF}}$
$\mathcal{L}_{OP}$
$\mathcal{L}_{\text{MF}}$
$\boldsymbol{Net}_{\text{SEG}}$
Segment Static
Segment Dynamic
$\boldsymbol{w}$

Figure 3.1: Overview of network architectures used in our experiments. The top panel shows conventional (*baseline*) approach that directly predicts 6DoF camera motion ($\mathcal{L}_{cam}$). The middle panel displays our proposed *single layer model* which predicts ego motion assuming a static (rigid) environment. We train the model with additional unsupervised losses based on optical flow ($\mathcal{L}_{OF}$), motion field ($\mathcal{L}_{MF}$), and orthogonal projection ($\mathcal{L}_{OF}$) described in Section 3.1.1. Our model supports both supervised (red) and unsupervised (green) losses during training. The bottom panel shows a *two layered model* variant that segments a scene into static and dynamic components and only uses static component for camera motion prediction. When input depth is not available, we utilize an additional monocular depth estimation network to predict it.

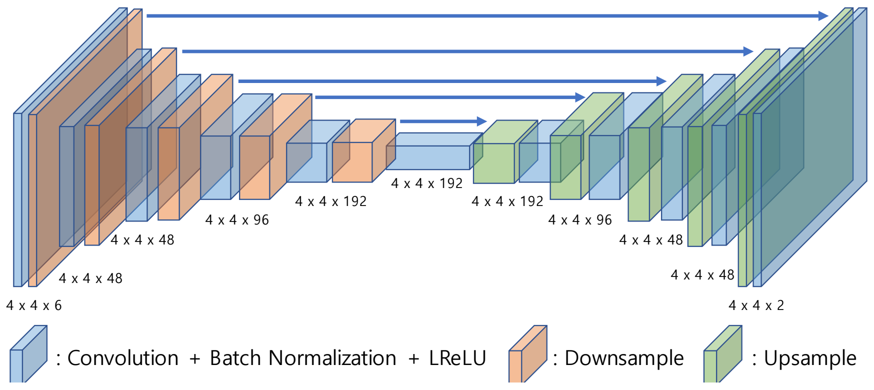
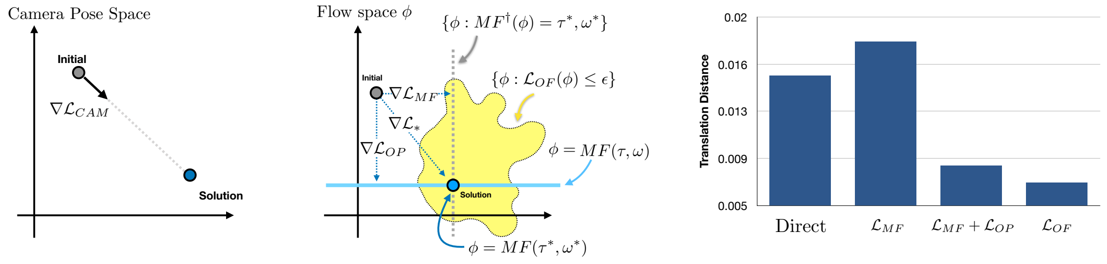
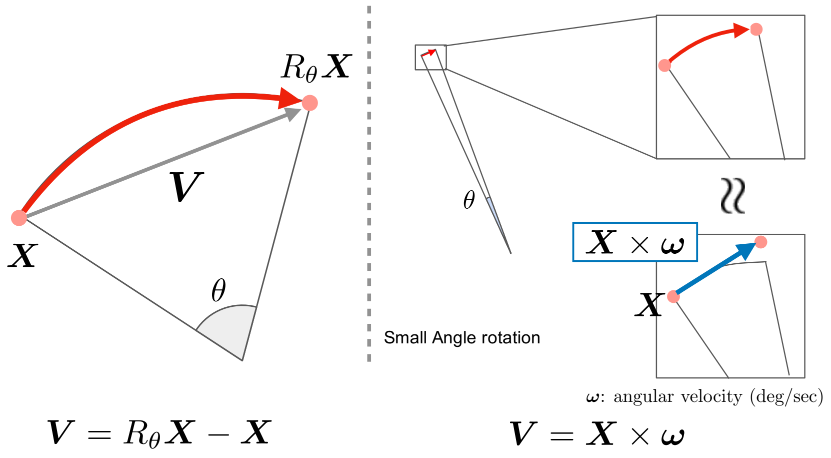


Figure 3.2: Detail network architecture of Net_{OF} and Net_{Seg}.



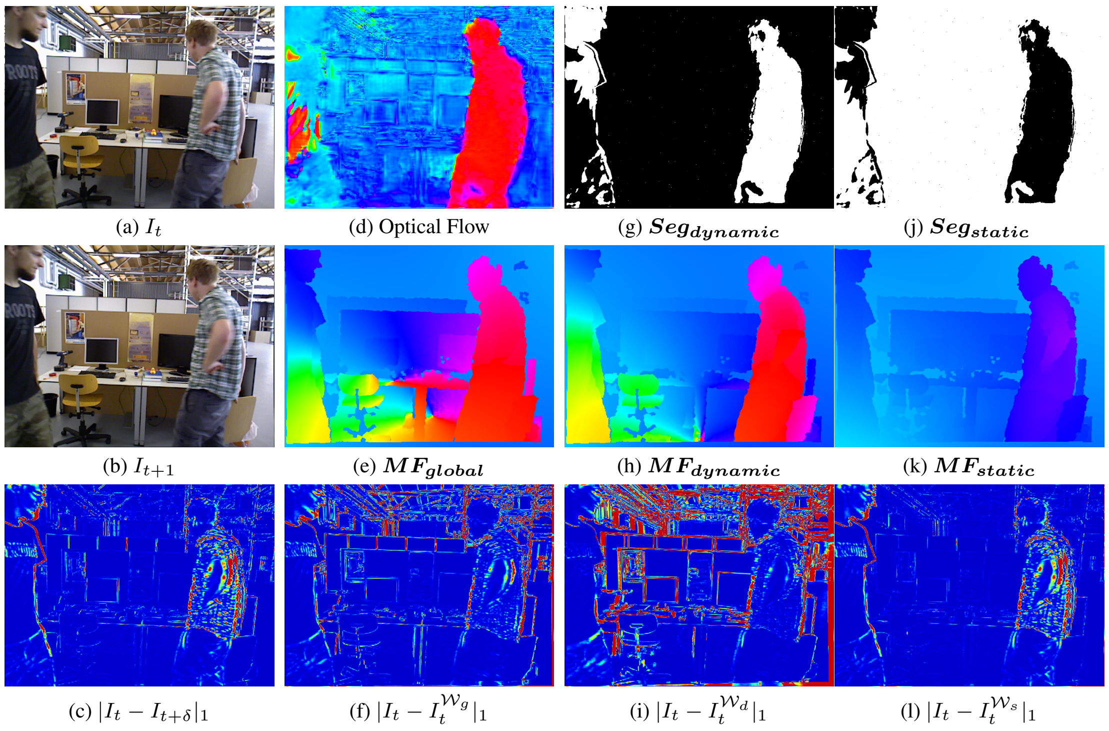
(a) Predict pose directly (b) Predict pose via flow space (c) losses comparison

Figure 3.3: Schematic interpretation of different loss functions. (a) Supervised training of direct models utilize a loss defined on camera pose space. (b) Our approach defines losses on the space of pixel flows and considers losses that measure the distance to the true motion field, the sub-space of possible ego-motion fields (blue), and its orthogonal complement (gray dashed). The model is also guided by photometric or scene-flow consistency between input frames (yellow) (c) shows prediction error for supervised models trained with different combinations of these losses and indicates that using losses defined in flow-space outperforms direct prediction of camera motion.



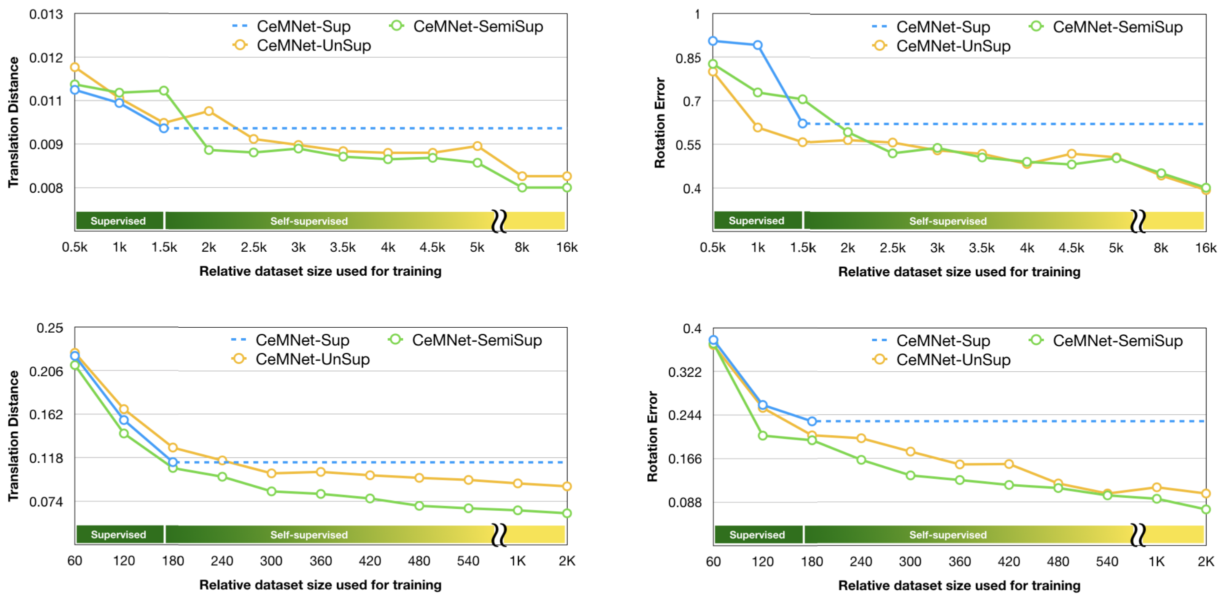
$R_\theta \boldsymbol{X}$
$\boldsymbol{V}$
$\boldsymbol{X}$
θ
θ
Small Angle rotation
$\approx$
$\boldsymbol{X} \times \boldsymbol{\omega}$
$\boldsymbol{X}$
ω: angular velocity (deg/sec)
$\boldsymbol{V} = R_\theta \boldsymbol{X} - \boldsymbol{X}$
$\boldsymbol{V} = \boldsymbol{X} \times \boldsymbol{\omega}$

Figure 3.4: Illustration of continuous formulation. With Euler format of rotation, motion vector from a point X to transformed point $R_\Theta X$ is its difference $V = R_\Theta X - X$. In contrast, if we use continuous formulation, we can define the motion as a cross product of X and its angular velocity ω. With small enough angle Θ, the motion vector can be approximated.



(a) I_t
(d) Optical Flow
(g) $\boldsymbol{Seg_{dynamic}}$
(j) $\boldsymbol{Seg_{static}}$
(b) I_{t+1}
(e) $\boldsymbol{MF_{global}}$
(h) $\boldsymbol{MF_{dynamic}}$
(k) $\boldsymbol{MF_{static}}$
(c) $|I_t - I_{t+\delta}|_1$
(f) $|I_t - I_t^{\mathcal{W}_g}|_1$
(i) $|I_t - I_t^{\mathcal{W}_d}|_1$
(l) $|I_t - I_t^{\mathcal{W}_s}|_1$

Figure 3.5: A sample result on a dynamic sequence from TUM [100]. From an input frame pair (a) and (b), $\boldsymbol{Net}_{OF}$ predicts optical flow (d). Both camera and object motion are visible in the frame difference (c). A single motion field (e) is dominated by large object motions and yields poor warping error (f), particularly on the background. Our model includes a segmentation network $\boldsymbol{Net}_{seg}$ that divides the image into dynamic and static masks (g,j) and fits corresponding motion fields (h,k). These provide better warping error on the objects (i) and background (l) respectively.



(a) Translation Error

(b) Rotation Error

Figure 3.6: Camera motion error on held-out test data as a function of training set size for TUM (top) and Virtual KITTI (Bottom) RGBD datasets. The blue line denotes training a supervised model that can't exploit unlabeled data. Introducing self-supervised warping losses yields much better performance when either using only unsupervised training (yellow) or semi-supervised training (green). Surprisingly, unsupervised training is actually competitive with supervised training for estimating rotation (b) but performs worse for translation (a).

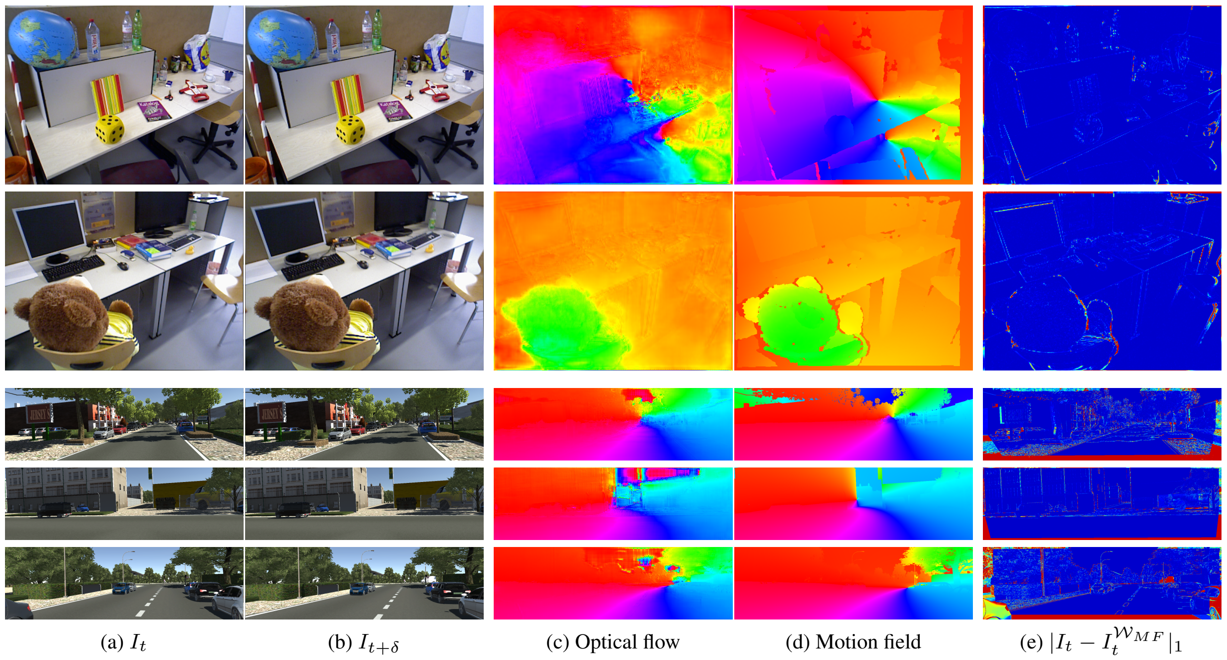


Figure 3.7: Visualizations of our single layered model. Top three rows come from TUM [100] dataset and bottom three come from Virtual KITTI [25]. From the input images (a) and (b), the predicted flow, and recovered motion field are displayed in (c) and (d) respectively. Since motion field is derived from camera pose estimate, the error between I_t and motion field based warped image $I_t^{\mathcal{W}_{MF}}$ reflects the accuracy of predicted camera motion. If the predicted camera pose and depth is ideal, then the error in (e) should be zero.

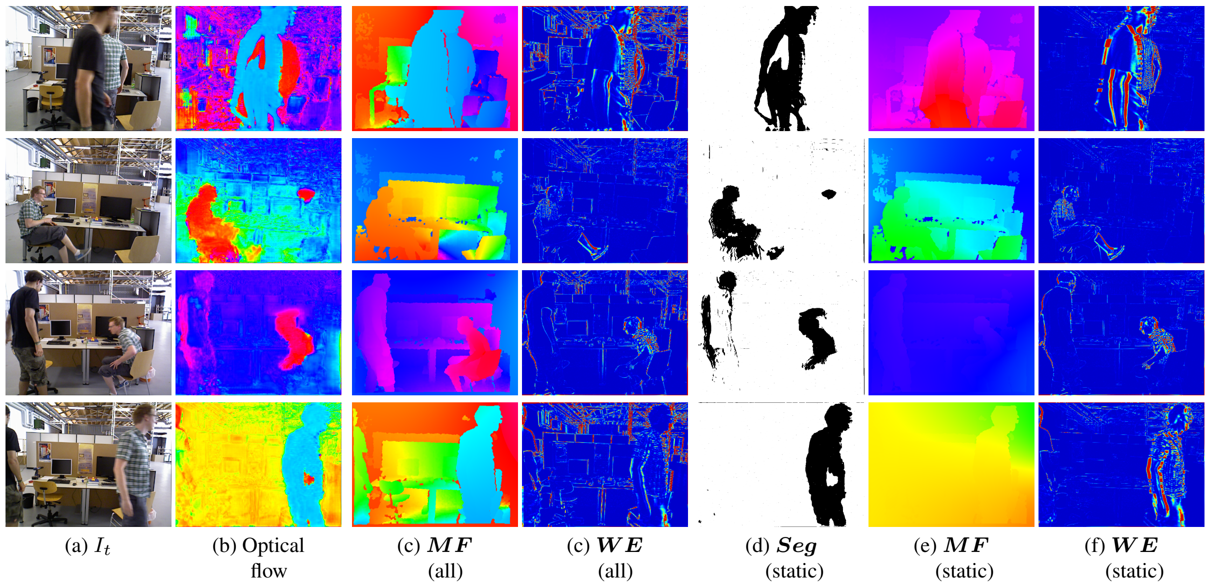

(a) I_t
(b) Optical flow
(c) $\boldsymbol{MF}$ (all)
(c) $\boldsymbol{WE}$ (all)
(d) $\boldsymbol{Seg}$ (static)
(e) $\boldsymbol{MF}$ (static)
(f) $\boldsymbol{WE}$ (static)

Figure 3.8: Intermediate results of two layered model for dynamic scene camera pose prediction. Without separating static and dynamic components, it is difficult to get good camera motions (high error in (c)). However, as shown in (f), it is possible to predict camera motion for background by fitting only the static segment (d).

Chapter 4

Self-supervised learning for better post refinement

For the movable agents which have visual data obtaining system (*e.g.* eye, camera), recognizing their pose through visual information is a critical task in order to understand their surroundings and plan the next motion. In computer vision and robotics, the task is called visual ego-motion prediction or visual odometry and has been widely studied. Many recent learning based ego-motion prediction methods [126, 32, 65, 123] show successful results on car driving dataset (*i.e.* KITTI). Specifically, they focus on understanding ego-motion through short-term visual data and try to develop better and better loss functions to boost the local accuracy. Are those enough to achieve complete scene reconstruction? One of major difference between those learning based method and classic methods is whether they use global pose refinement. Since it is difficult to implement global refinement as differentiable layers, learning-based models tend to exclude the refinement step. In this chapter, we focus on the bridge between local estimates and global refinement by predicting valuable clues to do better pose refinement.

The pose prediction task is a part of simultaneous localization and mapping (SLAM) and especially

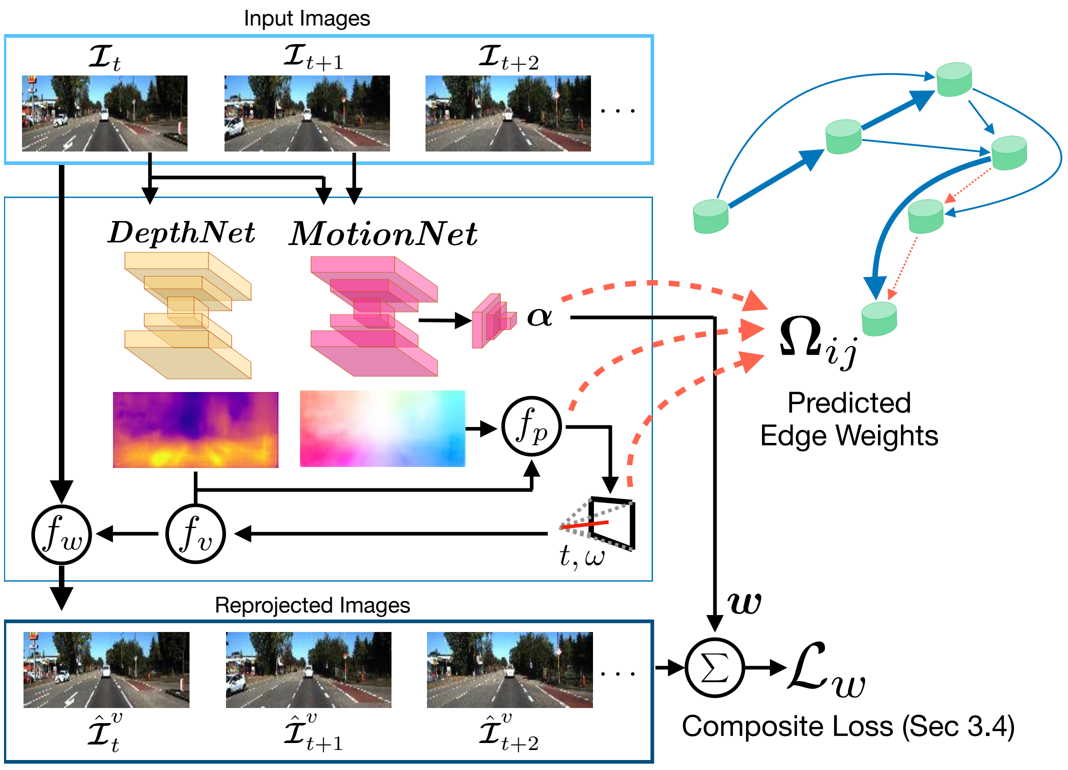


Figure 4.1: The flow to get reliability score. We combine camera ego-motion driven warp images with trainable weight α. From this loss, we train reliability score between frames and that reflects weights of each predicted ego-motion(Black means training flow and orange flow is for inference). See Section 4.1.4 for detail.

the local prediction is called front-end part of SLAM. As its name implies, SLAM does localization and mapping at the same time. In the localization step, it utilizes the generated map. Without using mapping part, dead-reckoning would make large drift error [11] because the map helps the agent to reset the accumulated error by aligning the agent to generated map.

Even though utilizing a map has the critical advantage, many modern learning based visual odometry (VO) systems do not include map creation and corresponding localization. Instead of using a map for localization, global pose refinement can be an alternative to reset accumulated error. Li *et al.*[67] use global pose refinement after they predict local estimates in order to achieve global adjustment effects. After the global refinement, the accumulated error is greatly reduced. In the adjustment stage, they put equal weight on pair-wise relative estimates. However this has some limitation when some estimates have serious error for example false-positive closed loop. When

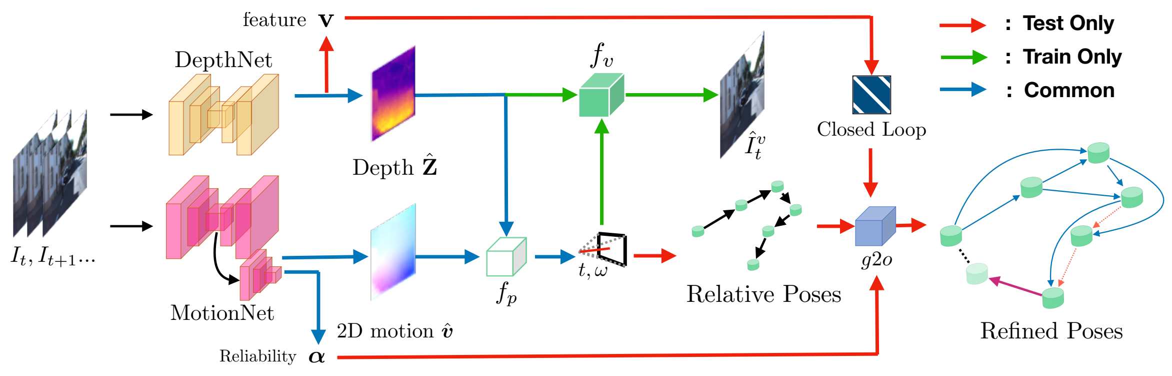


Figure 4.2: The overview of our training scheme. We train depth, 2D motion, and reliability score as well. In testing time, we use feature vector $\boldsymbol{v}$ for loop detection, and reliability score driven information matrix for post global refinement.

we train and test on a limited size of dataset, it rarely happens but if a learned model runs in real environment, it could be common case. If we have multiple predictions between edges and choose good quality amongst them, then it helps to avoid serious edge errors.

In this chapter, we propose incorporating training architecture to train monocular depth and camera pose derived from 2D velocity. Moreover, since we consider global optimization , we predict optimal information matrix (reliability) to get improved global optimization results. The results are demonstrated with several publicly available monocular video dataset and evaluated. The itemized contributions are as follows:

- We suggest a novel scheme for depth and camera pose formulated with angular velocity formulation.
- In order to estimate local reliability, we use the first order derivatives of loss function with respect to camera pose as well as learned reliability score from composite loss.
- We model the relationship between predicted camera pose and reconstruction loss to efficiently refine local estimates.

4.1 Method

4.1.1 Overview

As displayed in Figure 4.2, our architecture includes two separated U-Net shaped networks [92, 125] for 2D motion and depth prediction respectively. Specifically, ***DepthNet*** predicts depth and scene feature vector $f_{\boldsymbol{\theta}}^{Depth}(\boldsymbol{\mathcal{I}}_i) \rightarrow \hat{\boldsymbol{Z}}_i, \mathbf{v}_i$, and ***MotionNet*** predicts 2D motion flow (i.e. optical flow) and reliability score $f_{\boldsymbol{\theta}}^{Motion}(\boldsymbol{\mathcal{I}}_i, \boldsymbol{\mathcal{I}}_j) \rightarrow \boldsymbol{\mathcal{OF}}_{ij}, \alpha_{ij}$. Please note that we use bold font for vector while normal font for single scalar. Those two networks are jointly trained in the self-supervised manner. To compute 6 DoF camera pose, we use solve linear optimization from predicted depth and 2D motion. All network layers are fully differentiable so we can do end-to-end training. For the detail of training, please refer to Section 4.1.2.

4.1.2 Baseline self-supervised learning model

Since we use a sequential pair of RGB frames as input, it is possible to use one side image as the supervisory signal to the other side of image as used in many other self-supervised learning scheme [126, 32] by leveraging the geometric constraint between the pair. Similar to [91, 123, 63], we use photometric consistency loss from relative 2D motion for self-supervision. In our training setting, by feeding a pair of images $(\boldsymbol{\mathcal{I}}_i, \boldsymbol{\mathcal{I}}_j)$ to ***MotionNet***, we first predict 2D pixel-wise flow $\boldsymbol{\mathcal{OF}}_{ij} \in \mathbb{R}^{H \times W \times 2}$, then from a source image I_i, we synthesize target image $f_w : (\boldsymbol{\mathcal{I}}_i, \boldsymbol{\mathcal{OF}}_{ij}) \rightarrow \hat{\boldsymbol{\mathcal{I}}}^o_{i \rightarrow j}$ by utilizing spatial transformer network (STN) [49]. Then we can simply design the loss by measuring the distance between target image $\boldsymbol{\mathcal{I}}_j$ and warped image from the source $\hat{\boldsymbol{\mathcal{I}}}^o_{i \rightarrow j}$.

4.1.3 2D motion field driven scene synthesis

While other self-supervised VO systems use Euler transformation for scene synthesis, [126, 32] we utilize angular velocity as used in [41, 52, 63]. By explicitly utilizing the 2D velocity, we can take further advantages such as semantic segmentation to filter out unnecessary objects or refinement from 2D motion while direct pose regression cannot. According to pinhole camera model, the relationship between a 2D coordinate x and corresponding 3D coordinate X as a function of time (t) is written as:

$$x(t), y(t) = \frac{fX(t)}{Z(t)}, \frac{fY(t)}{Z(t)},$$

where f means focal length. In order to get the 2D velocity of them, by taking a partial derivation for both sides with respect to time (t), we can get following equation [41, 52, 63]:

$$\boldsymbol{v} = \frac{1}{\mathbf{Z}}\mathbf{A}\boldsymbol{\tau} + \mathbf{B}\boldsymbol{\omega},$$

where matrix $\mathbf{A}$, and B are constant matrix which is cross product form with the focal length f, and $\boldsymbol{\tau} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ are respectively translation and angular velocity written in axis representation.

With given depth ($\mathbf{Z}$) and 6 DoF camera pose ($\boldsymbol{\tau}$, $\boldsymbol{\omega}$) we can simply compute rigid 2D motion (*i.e.* motion field). Let f_v is a function to generate the motion as: $f_{\boldsymbol{v}} : (\mathbf{Z}, \boldsymbol{\tau}, \boldsymbol{\omega}) \rightarrow \boldsymbol{v}$. Also, by solving linear system, we can define inverse operation as well $f_p : (\boldsymbol{v}, \boldsymbol{Z}) \rightarrow \boldsymbol{\tau}, \boldsymbol{\omega}$ [63]. One advantage of this operation is that we can build end-to-end differentiable network for converting optical flow to camera pose and vice versa. Similar to optical flow based scene synthesis, we generate camera pose driven warp image with 2D warping function f_w as:

$$\hat{\boldsymbol{\mathcal{I}}}_t^v = f_w(\boldsymbol{\mathcal{I}}_s, \boldsymbol{v})$$

By minimizing the distance $D(\cdot)$ between $\boldsymbol{\mathcal{I}}_t$ and $\hat{\boldsymbol{\mathcal{I}}}_t^v$, we can train 2D motion and ego-motion as well as depth in a single loss function.

Appearance based loss In order to achieve robust image comparison, we adapt SSIM [113] and L_1 mixed image comparison ($\lambda = 0.85$) method used in [5].

$$D(a, b) = \lambda \frac{1 - SSIM(a, b)}{2} + (1 - \lambda)||a - b||_1$$

Our baseline self-supervised photometric consistency loss can be described as below:

$$\begin{aligned} \mathcal{L}_o &= D(\boldsymbol{\mathcal{I}}_j, \hat{\boldsymbol{\mathcal{I}}}_{i \to j}^o) \\ \mathcal{L}_m &= D(\boldsymbol{\mathcal{I}}_j, \hat{\boldsymbol{\mathcal{I}}}_{i \to j}^v) \end{aligned} \tag{4.1}$$

4.1.4 Multi-pair composite warping loss

One the simplest way of describing an image frame without using itself is the warping version of the very previous frame $\boldsymbol{\mathcal{I}}_{t+1} = f_w(\boldsymbol{\mathcal{I}}_t, \boldsymbol{v}_{t\to t+1})$. However, it is also possible to describe the frame as a weighted combination of multiple previous images as $\hat{\boldsymbol{\mathcal{I}}}^c_t = \sum_i \alpha_i f_w(\boldsymbol{\mathcal{I}}_{t-\delta_i}, \boldsymbol{v}_{t-\delta_i\to t})$. Intuitively, the weight α_i indicates how much each neighbour contribute to describe $\hat{\boldsymbol{\mathcal{I}}}^c_t$. As displayed in Figure 4.1, with this weighted composite warping image, we can train a reliability score between frames. We build composite warping loss described as follows:

$$
\begin{aligned}
w_{st} &= \frac{e^{\alpha_{st}}}{\sum_{\cdot t\in E} e^{\alpha_{\cdot t}}} \\
\hat{\boldsymbol{\mathcal{I}}}^c_t &= \sum_{st\in E} w_{st}\hat{\boldsymbol{\mathcal{I}}}^v_{s\to t} \\
\mathcal{L}_c &= D(\boldsymbol{\mathcal{I}}_t, \hat{\boldsymbol{\mathcal{I}}}^c_t)
\end{aligned}
$$

where α_{st} is learned parameter which implies reliability between image $\boldsymbol{\mathcal{I}}_s$ and $\boldsymbol{\mathcal{I}}_t$. Also, w_{st} is normalized weight value using *softmax* from initial prediction α_{st}. Then, combined image is used for loss similar to previous losses.

4.1.5 Edge aware smoothness loss

In order to regularize smoothness, we use edge aware smoothness term in addition to photometric consistency loss as used in [32]. We apply the regularization for both predicted 2D flow and depth as:

$$\begin{aligned}
\mathcal{L}_{s.OF} &= \sum_{d\in\mathcal{D}} |\delta_d \boldsymbol{\mathcal{OF}}_t| e^{-|\delta_d \boldsymbol{\mathcal{I}}_t|}, \\
\mathcal{L}_{s.Z} &= \sum_{d\in\mathcal{D}} |\delta_d \hat{\boldsymbol{Z}}_t| e^{-|\delta_d \boldsymbol{\mathcal{I}}_t|}, \\
\mathcal{L}_s &= \mathcal{L}_{s.OF} + \mathcal{L}_{s.Z}
\end{aligned}$$

where $\mathcal{D}$ is a set of gradient directions such as $\{x, y\}$.

4.1.6 Pose cycle consistency loss

Since we deal with sequential dataset, we can build additional consistency loss about relative relationship to another relationship used by Li *et al.*[67]. Specifically, a merged relative transformation of $i \to j$ and $j \to k$ should be similar to a longer transformation of $i \to k$. It can be applicable on both camera pose and predicted optical flow as:

$$\begin{aligned}
\mathcal{L}_p &= \sum_{(i,j,k)\in\mathcal{E}} \left|\boldsymbol{T}_{ij}\boldsymbol{T}_{jk}\boldsymbol{T}_{ik}^{-1} - \mathbf{I}\right|_1, \\
i, j, k &\in \mathcal{E} \quad \textbf{if} \quad i < k \leq i + N, i < j < k,
\end{aligned}$$

where $\boldsymbol{T}_{ij} \in \mathbb{R}^{4\times x}$ is transformation matrix from i-th frame to j-th frame and N means maximum edge distance. After adding all losses, our final loss function is:

$$\mathcal{L} = \mathcal{L}_o + \mathcal{L}_m + \mathcal{L}_w + \lambda_p\mathcal{L}_p + \lambda_s\mathcal{L}_s.$$

4.1.7 Information matrix for global refinement

Theoretically, the result quality from the most ego-motion prediction methods eventually suffer from accumulation error. Even though each predicted relative ego- motion is good quality, unless prefect, error accumulates as the concatenated path becomes longer. Thus, it is necessary to apply refinement step to minimize the accumulated error.

Amongst possible optimization schemes, we choose pose graph optimization and use *g2o* framework implementation [61] with python wrapper. In order to do pose graph optimization, we use two things: initial pose (**X**) and relative relationship between poses (**p**). According to Li *et al.*[67], they consider the relative edges between nodes with same weights for the refinement step. However, in practice, each relative relationship does not equally contribute the good pose graph. More reliable edges should get higher weight than less reliable edges.

According to well studied graph optimization [38, 61], the framework minimizes following energy function:

$$\begin{aligned} \mathbf{E} &= \sum_{i \in E} \mathbf{e}(\mathbf{X}_i, \mathbf{X}_j, \mathbf{p}_{ij})^\top \mathbf{\Omega}_{ij} \mathbf{e}(\mathbf{X}_i, \mathbf{X}_j, \mathbf{p}ij) \\ \mathbf{X}^* &= \arg\min_{\mathbf{X}} \mathbf{E}, \end{aligned} \tag{4.2}$$

where $\mathbf{X}_i$ means i-th absolute position and $\mathbf{p}_{ij}$ is relative transformation from i to j-th camera position. Also, $\mathbf{\Omega}_{ij}$ is information matrix between i to j-th poses, which reflects the uncertainty between the frames. We focus on the information matrix in order to selectively consider more reliable pose prediction for pose this optimization step.

Building information matrix In pose graph optimization context, information matrix measures uncertainty of measurement and also known as inverse covariance of observed distribution [38].

Intuitively, the higher values an information matrix has, it effects more to the optimized poses. In contrast to other classical SLAM methods [105, 81, 21], it is difficult to achieve the map-measured information matrix especially in this self-supervised training system since we only predict 6 DoF camera ego-motion from a pair of input frames without creating map. Instead of achieving exact information matrix, we indirectly solve this problem by keeping the natural property of information matrix which is uncertainty about prediction.

Within the pose optimization framework, the probability of an estimation ($\hat{textbfp}$) can be described as Gaussian distribution with $\mathbf{p}_{gt}$ and Σ_g as mean and covariance respectively written as:

$$\begin{aligned} p(\hat{\mathbf{p}}) &\sim G(\mathbf{p}_{gt}, \Sigma_g) \\ &= \frac{1}{\sqrt{(2\pi)^6|\Sigma_g|}} e^{-\frac{1}{2}(\hat{\mathbf{p}}-\mathbf{p}_{gt})^\top \Sigma_g^{-1}(\hat{\mathbf{p}}-\mathbf{p}_{gt})}. \end{aligned} \tag{4.3}$$

The inverse covariance matrix Σ_g^{-1} is corresponding to the information matrix mentioned earlier. In order to get the inverse covariance, we need to consider covariance matrix globally. At first, a covariance matrix of an estimation between i and j-th is an expectation of outer product of error vector itself as:

$$\begin{aligned} \Sigma_{ij} &= \mathbb{E}\big[(\hat{\mathbf{p}}_{ij} - \mathbf{p}_{ij}^{gt})(\hat{\mathbf{p}}_{ij} - \mathbf{p}_{ij}^{gt})^\top\big] \\ &= \mathbb{E}\big[\mathbf{e}_{ij}\mathbf{e}_{ij}^\top\big] \end{aligned}$$

Then, we assemble global matrix Σ_g by using local covariance matrices as follows:

$$\Sigma_g = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \dots \\ \vdots & \ddots & \\ \Sigma_{N1} & & \Sigma_{NN} \end{bmatrix}.$$

Then take inverse to the global covariance matrix in order to get the inverse covariance matrix used in Equation (4.3).

$$\Sigma_{ij}^{-1} = (\Sigma_g)_{ij}^{-1}$$

Since it is tricky to set the constraints to get the global inverse matrix, we convert the problem simple. If we consider a single sample and assume that the error has zero mean, then we can approximate the inverse covariance as:

$$\begin{aligned} \sigma &= \sqrt{\frac{1}{N}\sum_i (\mathbf{e}_i - \mu)^2} \\ &= \sqrt{\mathbf{e}_i^2} = |\mathbf{e}_i| \\ \sigma^{-1} &\approx \frac{1}{|\mathbf{e}_i|} \end{aligned} \tag{4.4}$$

Under the natural constraint of information matrix $\mathbf{diag}(\mathbf{\Omega}) > 0$ which is inverse covariance of measurement distribution, from Equation (4.2), we can intuitively find a good information matrix by taking inverse of ground truth driven error vector $\mathbf{e}$ as:

$$\mathbf{\Omega} \propto \mathbf{e}^{-1}$$

According to this approximation, we can assemble diagonal information matrix using inverse of error vector as described below:

$$\begin{aligned}
\mathbf{e}_{ij} &= (\mathbf{X}_i \hat{\mathbf{M}}_{ij})^{-1} \mathbf{X}_j \\
\mathbf{\Omega}_{ij}^{0:3,0:3} &\propto (\mathbf{e}_{ij}^{0:3,3})^{-1} \\
\mathbf{\Omega}_{ij}^{3:6,3:6} &\propto (\mathbf{ToNormQuat}(\mathbf{e}_{ij}^{0:3,0:3}))^{-1}
\end{aligned}$$

$\mathbf{X}_i \in \mathbb{R}^{4\times4}$ means i-th absolute ground truth position and $\hat{\mathbf{M}}_{ij} \in \mathbb{R}^{4\times4}$ is predicted relative motion from i to j-th frame. In order to make problem simple, we consider the information matrix is a diagonal matrix and the rotation error is converted into normalized quaternion by following the implementation of *g2o*.

We estimate uncertainty from two features: *Reliability* and *Sensitivity*. High uncertainty comes from low reliability and high sensitivity. Fortunately, we train reliability score from composite warping loss in Equation (4.2). For the sensitivity score, we can indirectly measure it by taking derivatives of motion field loss function $\mathcal{L}_m$ with respect to predicted relative camera pose ($\hat{\mathbf{p}}$). Intuitively, if a pose has low sensitivity, then the predicted value is also trustful since the prediction

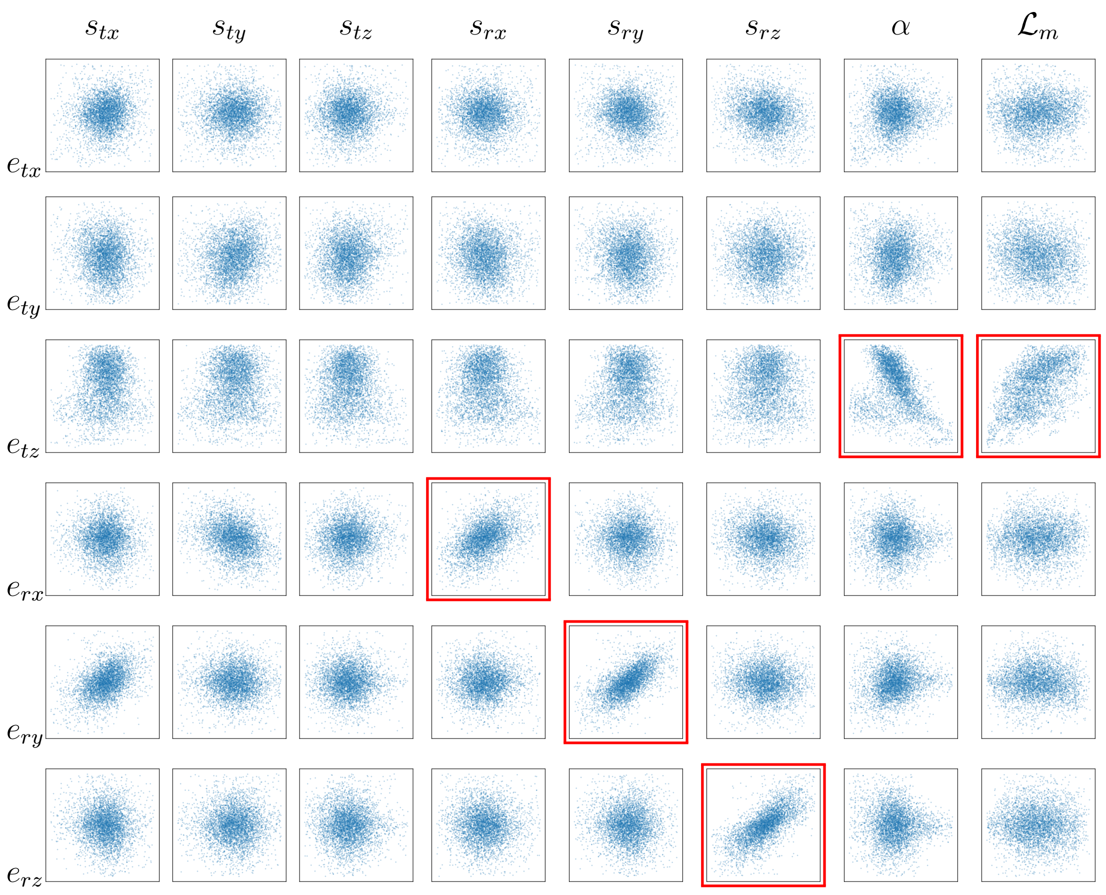


Figure 4.3: A pair-wise correlation examination between each axis of error vector and possible variables such as sensitivity, predicted α, and motion field loss $\mathcal{L}_m$. As marked with red frames, sensitivity show strong correlation with rotation part of error vector. In terms of translation, especially z-axis error, α and $\mathcal{L}_m$ show strong inverse and regular correlation.

is robust against fluctuation. We can describe the sensitivity as:

$$\boldsymbol{s}_{ij} = \frac{\partial \mathcal{L}_m(\hat{\mathbf{p}}_{ij})}{\partial \hat{\mathbf{p}}_{ij}} \tag{4.5}$$

In order to find variables which have correlation with error vector, we evaluated pair-wise correlation using KITTI dataset sequence 00 in Figure 4.3. Amongst the possible pairs, {rotation sensitivity ,rotation error} , {Reliability score and z-axis of translation error}, and {motion field

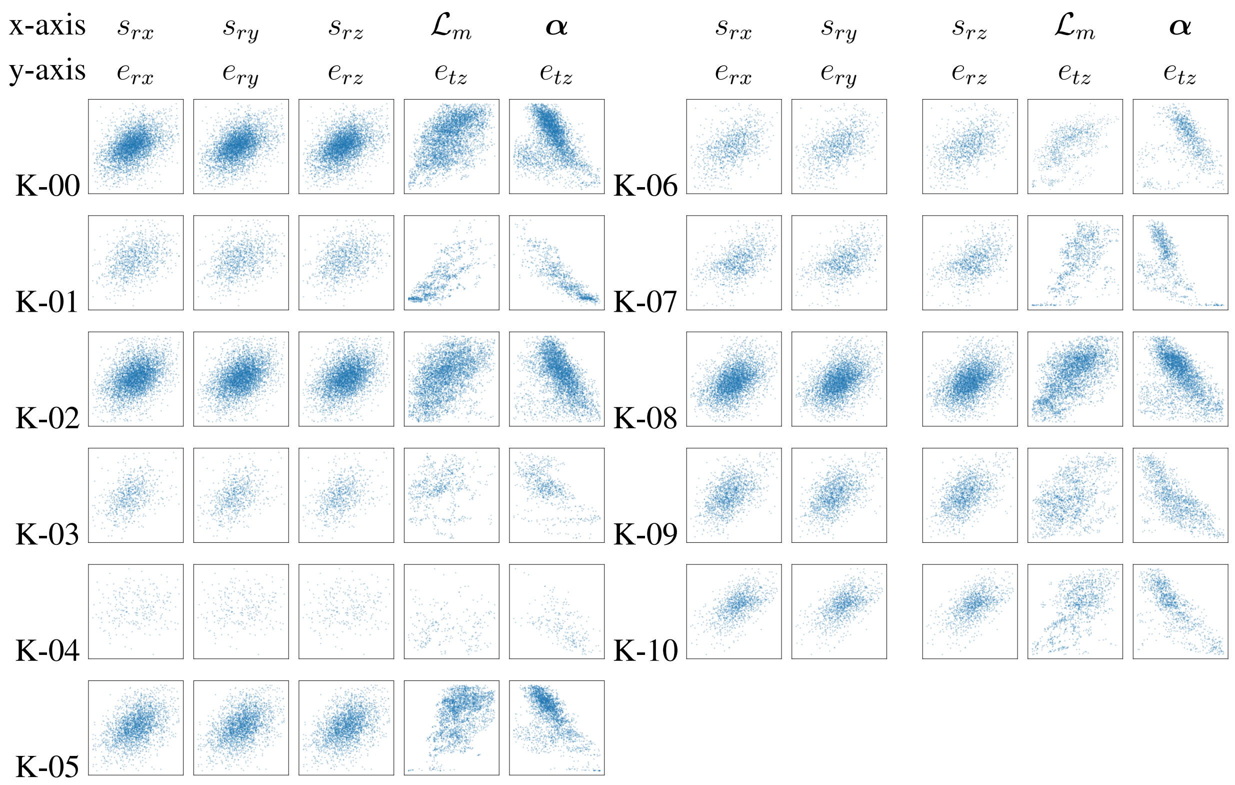


Figure 4.4: Scatter plot of the error vector elements versus uncertainty related variables (Sensitivity Equation (4.5), motion field loss Equation (4.1), and α (Section 4.1.4) using KITTI [28]. Rotation error and sensitivity show strong correlation between them, and translation error and motion field loss and α show strong correlation.Note that displayed variables are normalized for better visualization.

loss and z-axis of translation error} show strong correlation.

In order to verify the correlation across every KITTI dataset, in Figure 4.4, we plot the correlation between error motion ,and sensitivity, reliability score and motion field driven warping error ($\mathcal{L}_m$). The KITTI dataset shows strong correlation between sensitivity and rotation error while UZH-Drone dataset much weaker correlation. When a scene has high depth, then it suffers more with rotation error, and KITTI dataset tends to have higher depth than UZH-Drone dataset. According to the visual analysis, we can find proportional relationship between error motion and uncertainty variables. Then we can infer the property of good information matrix for KITTI dataset as:

$$
\begin{aligned}
\boldsymbol{\Omega}^r_{ij} &\propto \boldsymbol{s}^{-1}_{ij} \\
\boldsymbol{\Omega}^t_{ij} &\propto \alpha_{ij}, \mathcal{L}_m(\boldsymbol{p}_{ij})^{-1}
\end{aligned}
$$

Please note that the correlation heavily depends on the dataset characteristics. For the same types of dataset, they may share same correlation but need to investigate the correlations for totally different type of dataset such as under water exploring dataset.

In Figure 4.5, we evaluate several different types of information matrices on KITTI dataset. As the baseline, we set the constant weight on every edges, and use motion field loss, trained $\boldsymbol{\alpha}$, and camera pose gradient. Also, we tested the mixed version of three methods described before and put the best performance as optimal. The information matrix based on the inverse error (which uses ground-truth) blue color bar in the figure and shows the best performance. On average, our ground-truth free information matrix shows **15.54**% and **8.71**% performance boost in terms relative rotation and translation respectively. With ground truth driven information matrix, they show **27.45**% and **11.79**% improvement.

4.1.8 Numerical Analysis of Reconstruction loss

As described in Section 4.1.3, the motion field reconstruction loss $\mathcal{L}_m$ reflects the quality of camera pose and predicted depth $\hat{\boldsymbol{Z}}$. Chen *et al.*[13] use gradient with respect to loss in order to optimize the pose prediction without additional network predictions. With a given good depth prediction $\boldsymbol{Z}$, the probability of camera ego-motion is approximated similar to Gaussian distribution which is maximized at optimized camera pose $\mathbf{p}^*$ as:

$$
\begin{aligned}
p(\mathbf{p}_i|I_i, I_j, \boldsymbol{Z}) &\approx \frac{1}{\boldsymbol{N}} e^{-|I_i - f_w(I_j|\mathbf{p}_i, Z)} \\
&\approx \frac{1}{\boldsymbol{N}} e^{\frac{-|\mathbf{p}_i - \mathbf{p}_i^*|}{\boldsymbol{\sigma}}}
\end{aligned}
$$

By keeping the depth constant, given an axis of predicted camera pose $\hat{p}$, we can simplify the reconstruction loss as a function of camera pose. In order to do fine refinement of camera pose, we approximate the loss function as a quadratic function of camera pose as follows:

$$
\mathcal{L}(\hat{p}) \approx \frac{1}{2}(a\hat{p} - b)^2 + c,
$$

where a, b and c mean three coefficients of parabola equation. From the parabola function above, we can simply find the optimum camera pose $p^* = \frac{b}{a}$ which makes the gradient of loss function zero $\frac{\partial \mathcal{L}(\hat{p})}{\partial \hat{p}} = 0$, and those coefficients can be found by using the first and second derivatives of loss with respect to camera pose $\left.\frac{\partial \mathcal{L}(\hat{p})}{\partial \hat{p}}\right|_{\hat{p}} = a^2\hat{p} - ab$ and $\left.\frac{\partial^2 \mathcal{L}(\hat{p})}{\partial^2 \hat{p}^2}\right|_{\hat{p}} = a^2$. After some arrangement, we can find the optimal pose:

$$
\begin{aligned}
p^\dagger &= \arg\min_{\hat{p}} \mathcal{L}(\hat{p}) \\
&= \hat{p} - \frac{\partial \mathcal{L}(\hat{p})}{\partial \hat{p}} / \frac{\partial^2 \mathcal{L}(\hat{p})}{\partial^2 \hat{p}^2} \\
p^* &= \hat{p} - \lambda_n \frac{\partial \mathcal{L}(\hat{p})}{\partial \hat{p}} / \frac{\partial^2 \mathcal{L}(\hat{p})}{\partial^2 \hat{p}^2} \qquad (4.6)
\end{aligned}
$$

Since the optimal pose may sensitive to noise, for safety, we fuse the original pose and the optimal pose with balancing parameter λ_n. In Figure 4.6, we further analyze the effect of λ_n on the evaluation results. The absolute trajectory error (ATE) of KITTI sequences (00-08) shows minimum around $\lambda_n = 1.0$. Please note that with $\lambda_n = 0$ it is the original predicted pose and $\lambda_n = 1$ means optimal pose using the above optimization.

4.1.9 Close loop detection

We use K-Nearest Neighbors method to find closed loop instead of predefined ORB feature based bag-of-word (DBoW2 [26]) since the performance of pre-defined BoW is limited to its training set. For example, the ORB feature works well matched to car driving dataset such as KITTI [28], however it does not fit to another dataset which is recorded in different environment such as UZH Drone dataset [16].

We train the feature vector within our self-supervised training so that the closed loop detection is more adaptive and simpler to implement than local feature based BoW methods. However, since it is global feature vector of a scene, this feature shows clear limitation on partial matches. For example, in our experiments, we fail to detect opposite direction or cross intersection as a loop which require partial matching to find loop.

4.2 Implementation Details

In training, we use Adam optimizer [58] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Our model is implemented using Tensorflow 1.4 [1]. For close loop detection, we use Approximate Nearest Neighbour python wrapper. We use learning rate at the beginning 0.0001 and per each 5 epochs, we half the learning rate. For the final loss function, we use $\lambda_p = 0.015$ and $\lambda_s = 0.1$.

Method	**Seq.09**	**Seq.10**
ORB-SLAM (full) [81]	0.014 ± 0.008	0.012 ± 0.011
ORB-SLAM (short) [81]	0.064 ± 0.141	0.064 ± 0.130
DF-Net [128]	0.017 ± 0.007	0.015 ± 0.009
Godard *et al.*3 [33]	0.017 ± 0.008	0.015 ± 0.010
SFMLearner [126]	0.016 ± 0.009	0.013 ± 0.009
Klodt *et al.*[60]	0.014 ± 0.007	0.013 ± 0.009
EPC++(mono) [72]	0.013 ± 0.007	0.012 ± 0.008
GeoNet [123]	0.012 ± 0.007	0.012 ± 0.009
Struct2Depth [12]	$\mathbf{0.011 \pm 0.006}$	0.011 ± 0.010
GLNet [13]	$\mathbf{0.011 \pm 0.006}$	$\mathbf{0.011 \pm 0.009}$
Ours (W/O Refinement)	0.014 ± 0.008	0.013 ± 0.009
Ours (W/ Refinement)	0.0136 ± 0.008	0.0128 ± 0.009

Table 4.1: Average of absolute trajectory error (ATE) comparison with using KITTI odometry benchmark suite [28]. All method use monocular video for pose prediction and for comparison, we use 5-snippets in order to get absolute trajectory as introduced by Zhou *et al.*[126]

4.3 Experimental Results

In this section, we evaluate our model by focusing on the quality of predicted camera ego-motion. From the baseline which only use our consistency losses, we validate our extra features such as optimized information matrix post optimization, and other consistency losses. Also, we validate how the optimized pose (Equation (4.6)) helps to improve the result.

4.3.1 Dataset

The first dataset is KITTI [28] which is widely used by other geometric understanding task such as depth, ego-motion, and optical flow. Other interesting dataset is UZH drone dataset [16] that is relatively recently released dataset and has much different camera trajectories and scene statistics than for cars driving on a road.

4.3.2 Trajectory Evaluation

One common evaluation metric is absolute trajectory error (ATE) which measures absolute transformation error with limited frame snippets. Since our predicted translation has scale ambiguity, we do scale matching after collecting certain number of frames (common length is 5) then measure the error. In Table 4.1, we report ATE averaged over all 5 snippets as introduced in [126]. It is simple method to evaluate short-term prediction without considering scale ambiguity. Our results outperform traditional feature-based SLAM techniques and are on-par with other state-of-the-art learning based methods.

Another metric is relative translation/rotation error. For the comparison, we use two metrics t_{rel} and r_{rel}. t_{rel} means average relative translation (°/100m) from 100, 200, ... , 800 m, and r_{rel} means average rotation drift (100m) of 100, ... , 800 m. The result is displayed in Table 4.12. In order to make clear about ground truth based translation scaling, we tried both global scaling (†) and individual scaling (‡) as used in [5]. From the baseline which we only care about single edge between frames, we compare the result with better information matrix (I) and closed loop applied version (R) as well. Especially, the mark (*) means the result from ground truth driven information matrix. Our results generally shows better rotation error comparing to others.

Also, we tested our method using UZH dataset in Table 4.2. Since the dataset has low frame rate and relatively new feature at the input, it is difficult to get correct trajectories from feature based slam methods. In contrast, our method with closed loop show much low relative errors.

In Figure 4.7 we display qualitative results for KITTI dataset with closed loop refinement. Especially, we compared the baseline with equal weight for every edges, predicted weights and ground truth driven weights. Also, in Figure 4.8, we plot two sequences from UZH dataset. In the right side, we show separated error of xyz axis. With predicted weights, it shows slightly closer to the ground truth. In Figure 4.9, we plot the trajectories for the rest of sequences in UZH dataset.

Method	**Seq.03**†	**Seq.05**†	**Seq.07**†	**Seq.09**†	**Seq.06**‡	**Seq.10**‡
			t_{rel} relative translation			
ORB-SLAM2	12.58	10.13	12.31	14.94	13.39	15.52
Ours (baseline)	10.60	9.68	10.32	5.68	10.55	9.41
Ours + R	**3.62**	**4.52**	**5.94**	**2.70**	**6.55**	**7.97**
			r_{rel} relative rotation			
ORB-SLAM2	1.21	0.799	1.610	1.729	1.353	1.751
Ours (baseline)	0.33	0.31	0.79	0.92	0.81	0.66
Ours+ R	**0.22**	**0.20**	**0.45**	**0.31**	**0.45**	**0.61**

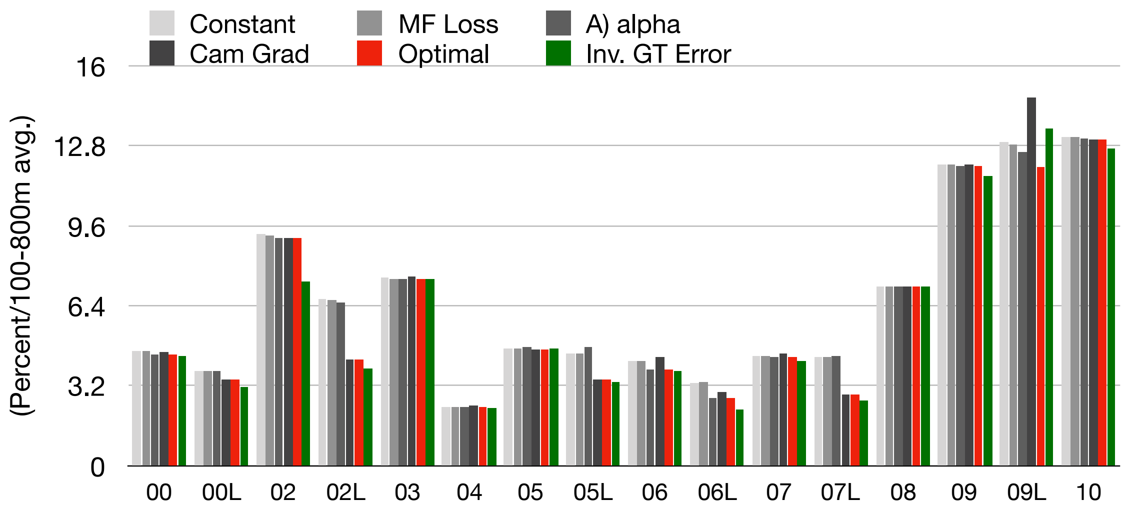
Table 4.2: Average of relative trajectory error (t_{rel}, r_{rel}) comparison with using UZH Drone racing dataset. According to the UZH evaluation metric, we use the average translation draft of $\{$40, 60, 80, 100, 120$\}$ meters. R states post refinement applied results.

4.3.3 Depth and motion prediction

In Figure 4.10, we display several example predictions from KITTI dataset. The predicted depth shows relatively strong signal at the edge part. We can understand that the edge parts contribute more for predicting good camera ego-motion and motion field. Some other example is in Figure 4.11 for UZH-Drone dataset. The dataset has a box at the center of racing area so the box is noticeable in terms of both predicted depth and motion.

4.4 Discussion

We have presented self-supervised integrated training network for 2D motion, depth and followed by camera ego-motion, and post global refinement. With photometric and cycle consistency loss, the reconstructed motion field helps to have more geometric consistency over entire image region. In order to boost post optimization step, we first analyze the good information matrix and then indirectly infer the good information matrix by using sensitivity and reliability score driven from composite warping loss. Even though we use incomplete loop closure method, the optimization results show good quality for KITTI and reasonable results on UZH-drone dataset. Moreover, through the numerical modelling of camera pose with respect to loss function, we show the effectiveness of local optimization. It is quite generally applicable to other camera pose prediction methods.



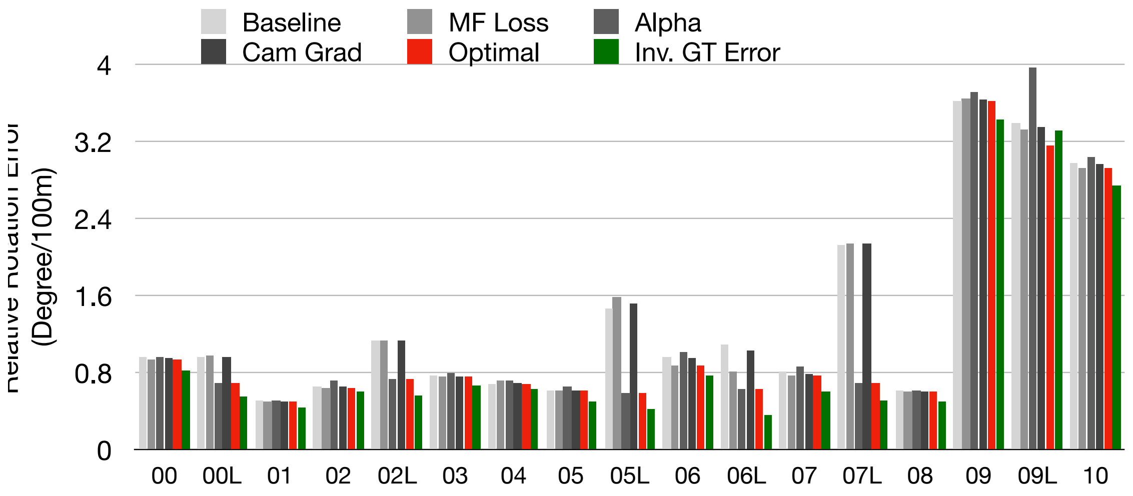


Figure 4.5: Result evaluation of various information matrix on KITTI dataset. We set constant information matrix as a baseline and then evaluate the results with several variations of information matrix. Generally, varying information matrix shows better results than the baseline. We display the best results as red color. Blue color is ground truth driven information matrix and shows best performance amongst them. Upper chart measures relative translational draft (%) and lower side measures relative rotational error (°). The letter L implies loop closure applied results. After loop closure, constant weights show unstable result but our optimized information matrix show relatively stable results.

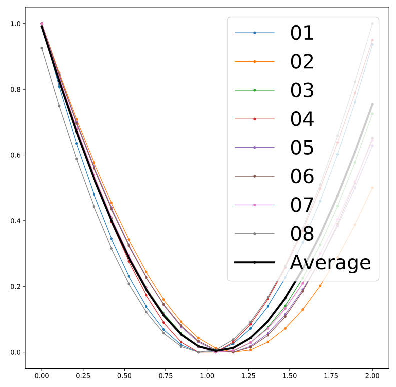


Figure 4.6: Absolute trajectory error (ATE) trend depends on λ_n. We measure how ATE (y-axis, normalized) changes with varying λ_n. X-axis means $\lambda_n = 0$ used in Equation (4.6). When $\lambda_n = 1$, the average ATE shows the minimum.

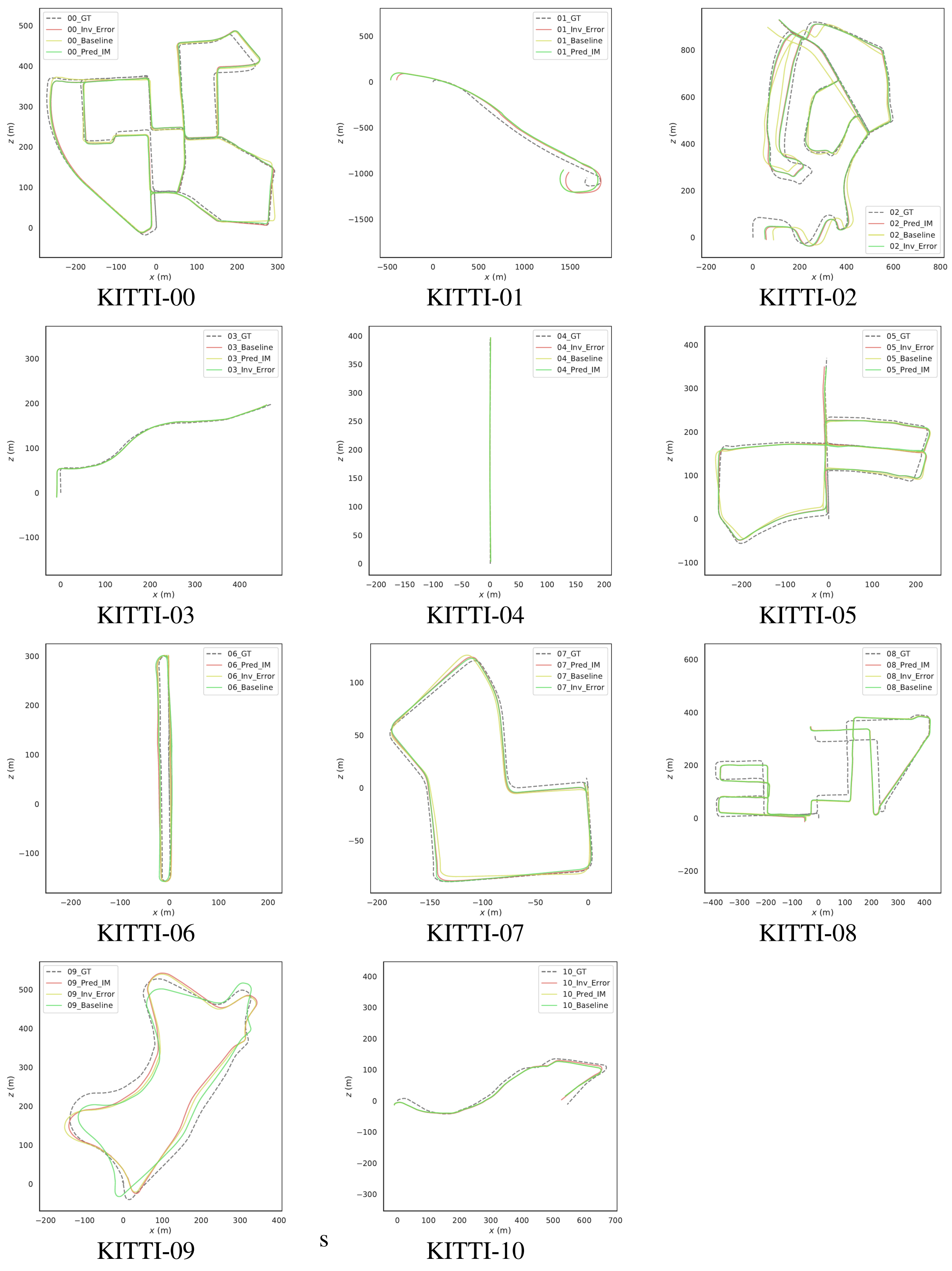
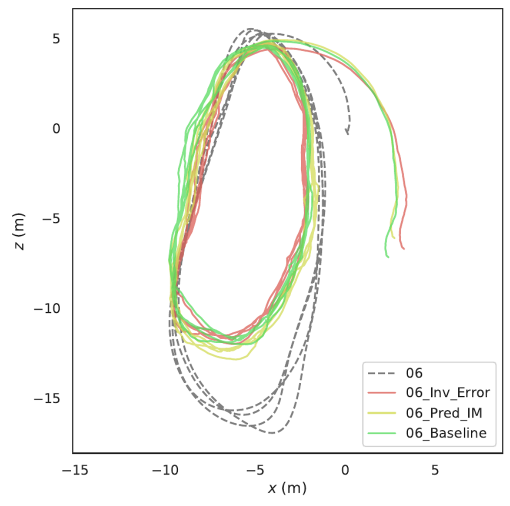
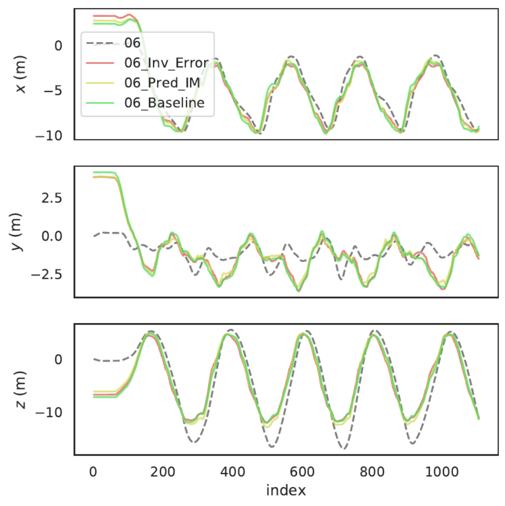


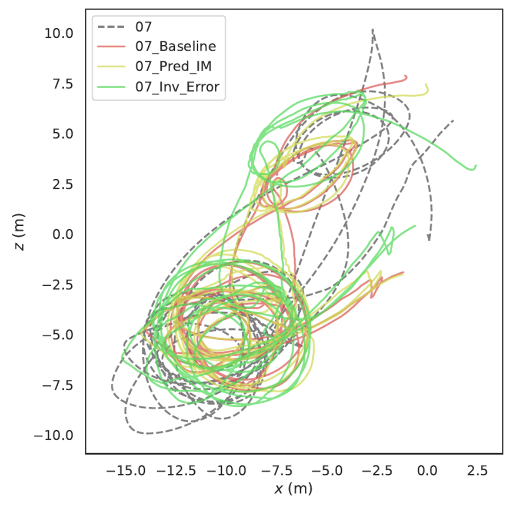
Figure 4.7: Qualitative trajectory results on selected sequences from KITTI with different types of information matrix: constant (baseline), predicted weight (PRED_IM), and ground-truth driven inverse error (Inv_Error).



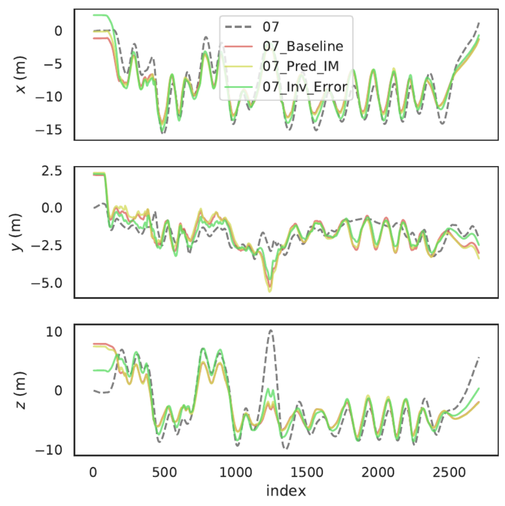
UZH-03



UZH-03-axis



UZH-07



UZH-07-axis

Figure 4.8: Qualitative trajectory results on selected sequences UZH-Drone dataset. Predicted trajectory comparison (left) and axis separated plot (right). Comparing to baseline, predicted information matrix and error driven show slightly improved results.

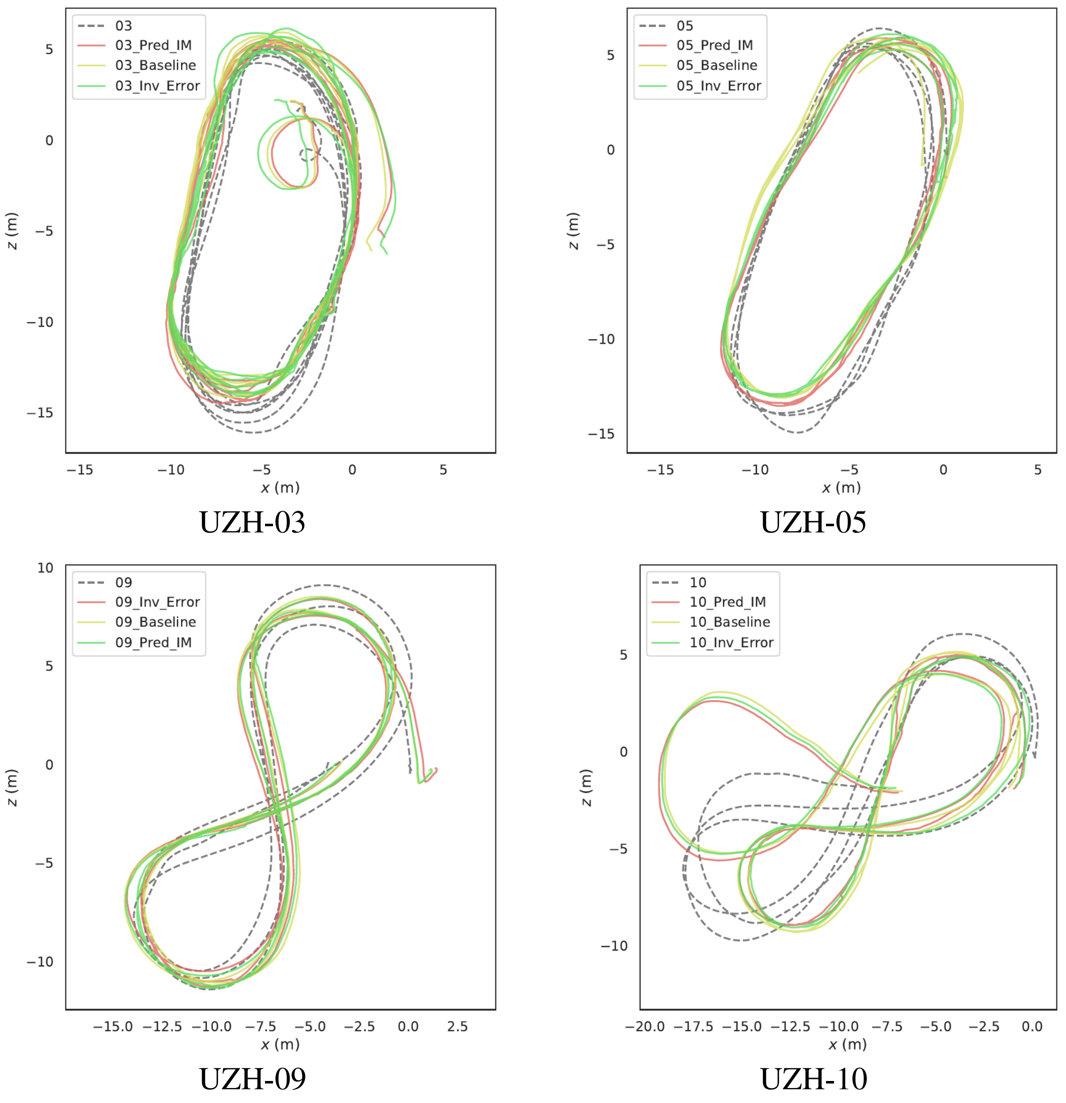


Figure 4.9: Qualitative trajectory results on selected sequences UZH-Drone dataset.

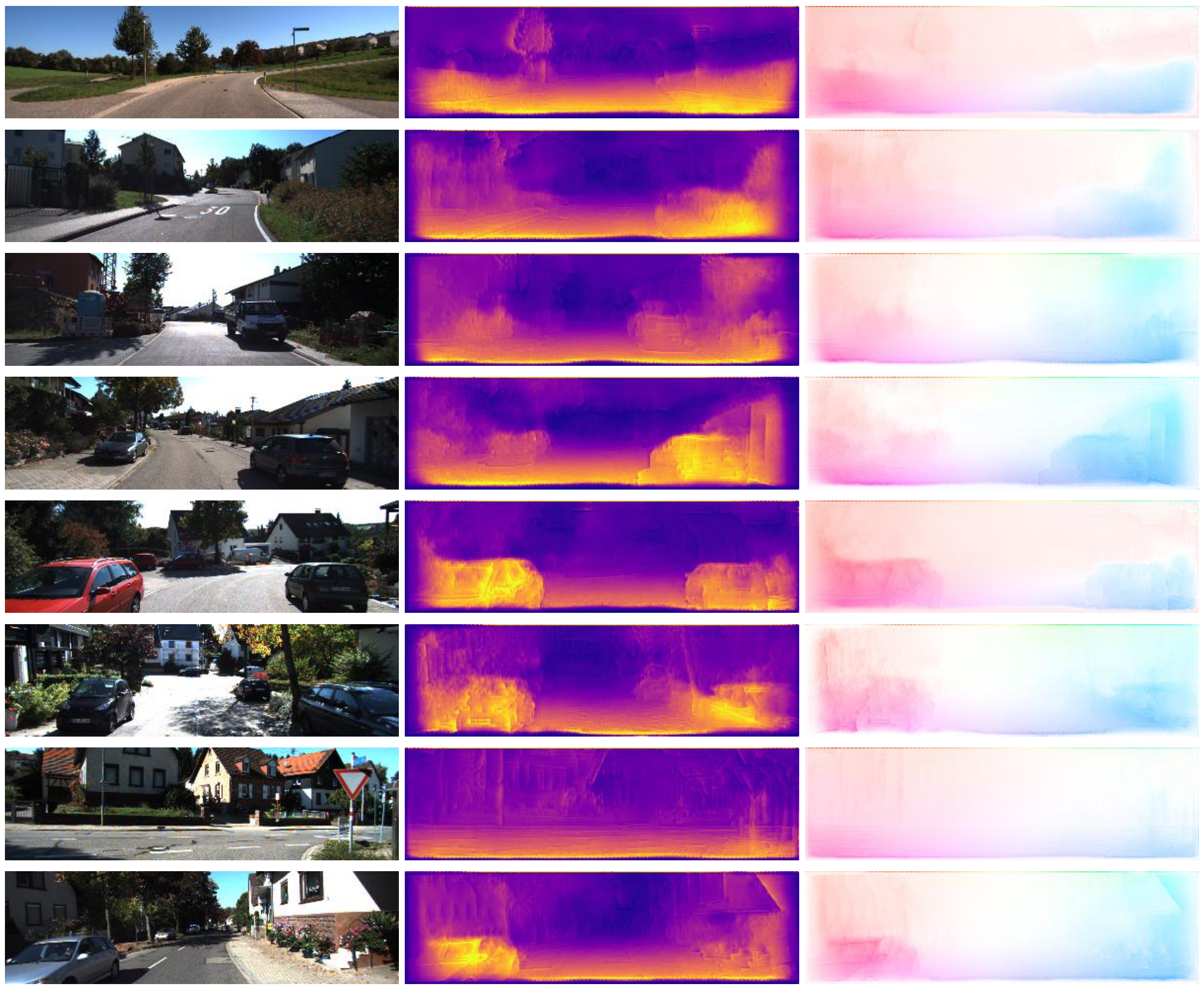


Figure 4.10: Qualitative results of depth prediction and motion using KITTI dataset (Sequence 09, not included in training set). From the input pair (a), we predict depth (b), and generated rigid motion field (c). Since we focus on ego-motion by minimizing motion field loss, our predicted depth emphasizes edge part which can help valid 2D motion and camera ego-motion.

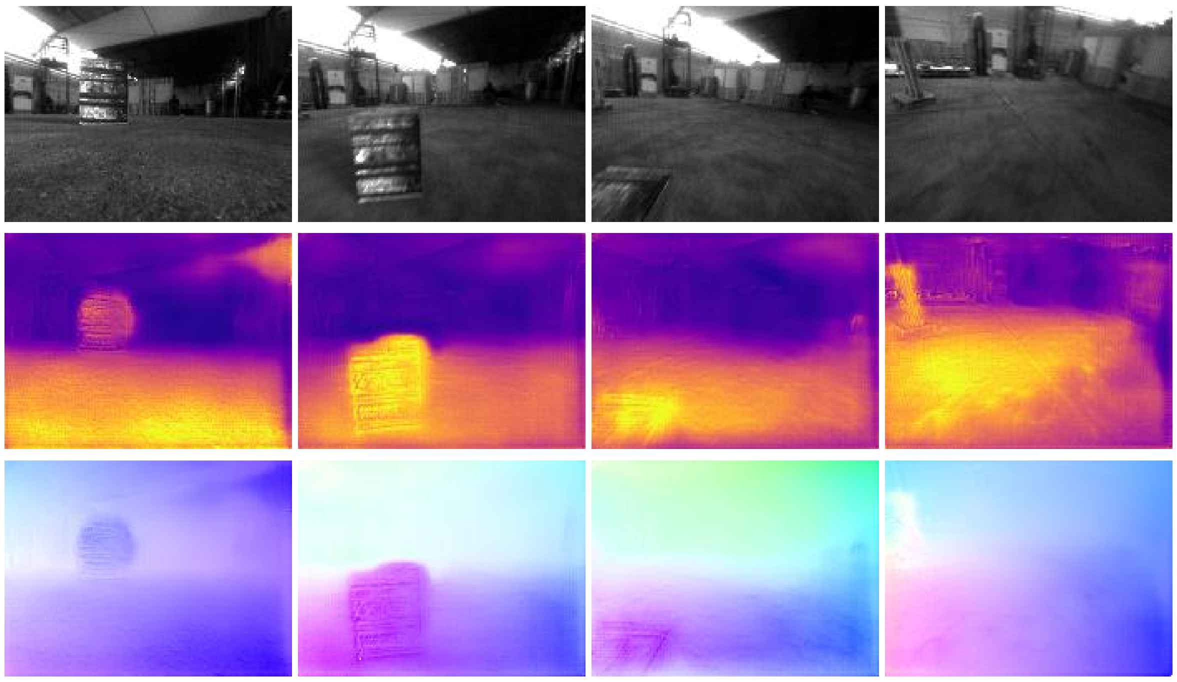


Figure 4.11: Results of depth prediction and motion of UZH-Drone dataset.

Method	**Input**	**Opt**	**S**	00*	01*	02*	03*	04*	05*	06*	07*	08*	09*	10*
						t_{rel} relative translation (Globally scaled)								
SFMLearner [126]	Mono	x	o	66.4	35.2	58.5	10.8	4.49	18.7	25.9	21.3	21.9	18.89	14.3
UnDeep VO [65]	Stereo	x	x	4.14	69.1	5.58	5.00	4.49	3.40	6.20	3.15	4.08	7.01	10.6
Zhu *et al.*[127]	Stereo	x	x	4.95	45.5	6.40	4.83	2.43	3.97	3.49	4.50	4.08	**4.66**	**6.30**
Ambrus[†] *et al.*[5]	Mono	x	o	4.88	**12.61**	4.19	**4.01**	3.2	5.26	8.18	6.33	7.34	6.72	9.52
Li *et al.*[67]	Stereo	x	x	4.33	17.98	6.89	4.51	**2.3**	3.91	4.6	3.56	**4.04**	8.10	12.9
Li *et al.*[67]	Stereo	o	x	3.24	17.98	4.85	4.51	**2.3**	**1.83**	2.74	3.53	**4.04**	6.23	12.9
Ours[†] (baseline)	Mono	x	o	4.60	55.7	9.26	7.50	2.34	4.70	4.20	4.39	7.17	12.0	13.1
Ours[†] + I	Mono	x	o	4.46	55.6	9.09	7.46	2.34	4.66	3.83	4.36	7.16	12.0	13.0
Ours[†] + I*	Mono	x	o	4.40	55.4	7.36	7.47	2.31	4.69	3.80	4.20	7.19	11.5	12.0
Ours[†] + IR	Mono	x	o	3.42	-	4.22	-	-	3.46	2.69	2.82	x	11.9	-
Ours[†] + IR*	Mono	x	o	**3.16**	-	**3.87**	-	-	3.35	**2.24**	**2.60**	x	13.5	-
						t_{rel} relative translation (Locally scaled)								
Ambrus[‡] *et al.*[5]	Mono	x	o	1.29	**1.63**	**1.06**	1.84	**0.55**	1.58	0.91	2.25	1.84	3.51	**2.32**
Ours[‡] (baseline)	Mono	x	o	2.20	1.98	2.52	1.53	1.22	1.35	2.54	1.36	1.68	8.47	6.98
Ours[‡] + I	Mono	x	o	2.07	1.92	2.40	1.47	1.22	1.33	2.27	1.28	1.65	8.48	6.81
Ours[‡] + I*	Mono	x	o	1.97	1.85	2.10	**1.41**	1.20	1.11	2.03	0.98	1.40	8.05	6.30
Ours[‡] + IR	Mono	x	o	**1.13**	-	2.08	-	-	1.10	0.88	0.86	x	6.33	-
Ours[‡] + IR*	Mono	x	o	1.30	-	1.86	-	-	**0.93**	**0.80**	**0.68**	x	10.7	-
						r_{rel} relative rotation								
SFMLearner [126]	Mono	x	-	6.13	2.74	3.58	3.92	5.24	4.1	4.8	6.65	2.91	3.21	3.30
UnDeep VO [65]	Mono	x	-	1.92	1.60	2.44	6.17	2.13	1.5	1.98	2.48	1.79	3.61	4.65
Zhu *et al.*[127]	Mono	x	-	1.39	1.78	1.92	2.11	1.16	1.2	1.02	1.78	1.17	1.69	1.59
Ambrus *et al.*[5]	Mono	x	-	0.55	0.48	**0.45**	0.94	**0.45**	0.67	**0.34**	1.15	0.70	**1.57**	**1.48**
Li *et al.*[67]	Stereo	x	-	1.85	1.44	2.61	2.82	0.87	1.64	2.85	2.39	1.53	2.81	3.17
Li *et al.*[67]	Stereo	o	-	1.35	1.44	1.60	2.82	0.87	0.7	2.6	2.02	1.53	2.11	3.17
Ours (baseline)	Mono	x	o	0.95	0.50	0.65	0.77	0.66	0.61	0.95	0.80	0.60	3.62	2.99
Ours + I	Mono	x	o	0.93	0.49	0.63	0.75	0.67	0.62	0.87	0.76	0.60	3.62	2.93
Ours + I*	Mono	x	o	0.81	**0.43**	0.59	**0.66**	0.62	0.49	0.76	0.59	**0.49**	3.43	2.74
Ours + IR	Mono	x	o	0.69	-	0.72	-	-	0.58	0.40	0.57	x	2.54	-
Ours + IR *	Mono	x	o	**0.54**	-	0.55	-	-	**0.41**	0.35	**0.51**	x	3.33	-

Figure 4.12: Evaluation results using KITTI dataset. We use two evaluation metric: t_{rel} : average translational root means squared error (RMSE) drift (%) from 100 to 800 m and r_{rel} : average rotational RMSE drift (°/100m) of length 100 to 800 m. * means results from ground-truth driven information matrix. **Opt** means post global optimization and **S** means do ground-truth driven scaling for comparison. For the clear comparison with translation scaling, † implies global scaling and ‡ means individual scaling. For the sequence 08, we fail to find closed loop so we marked as x.

Chapter 5

Space-time localization and mapping

A strategic question for scene understanding is how to leverage large repositories of images, video and other sensor data acquired over an extended period of time in order to analyze the content of a particular image. For static rigid scenes, a classic approach is to use visual SLAM, structure-from-motion, and multi-view stereo techniques to build up an explicit model of the scene geometry and appearance. These methods are well developed and have been scaled up to increasingly large problems in modeling outdoor and indoor scenes (see e.g., [2, 97, 24, 59, 23, 6]).

Such a geometric approach to scene understanding can make strong predictions about a novel test image including the camera pose (via feature matching and camera localization) and the appearance of points or surface patches projected into the image. However, reconstruction-based analysis typically neglects dynamic objects that change over time and are treated as outliers with respect to the estimation of a single rigid scene model. This problem becomes more acute as data is integrated over longer periods of time, during which an increasing proportion of objects in the scene may move non-rigidly. For example, people move on the time-scale of seconds while furnishings may shift on the time-scale of days and architecture and landscapes over years.

In this chapter, we investigate how the scope of such techniques can be extended by registering

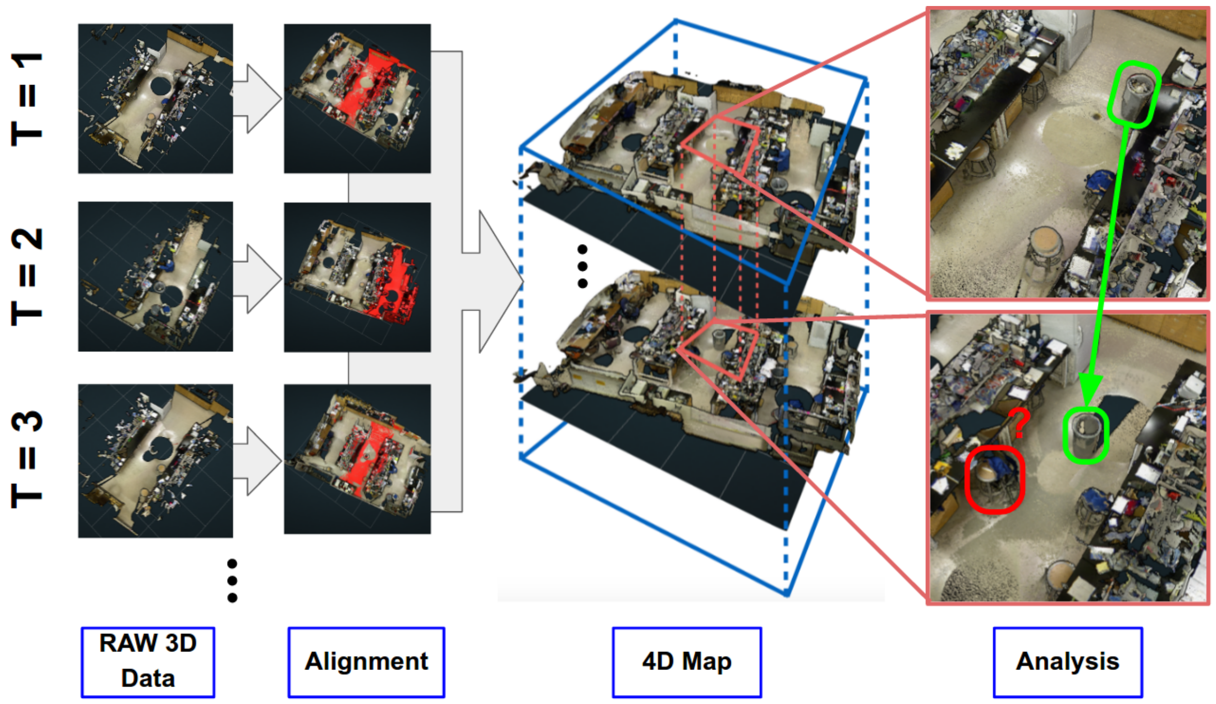


Figure 5.1: Raw measurements (pointclouds) captured at different times are aligned and probabilistically merged into a single 4D model that describes the spatio-temporal geometry of the scene. Model fitting reasons about occlusion, inferring complete surfaces even when they may not have been occluded at some times. The resulting integrated space-time map supports further analysis such as change detection and segmentation of dynamic objects.

observations to a 4D reconstruction that explicitly represents geometric changes over time. We focus specifically on space-time mapping of indoor scenes where RGB-D sensors provide streams of high-quality geometric data and odometry over short time intervals and limited fields of view, but data acquired, e.g. on different days, is inconsistent due to changes in the scene. The key inferential challenge is thus distinguishing sensor noise and reconstruction errors from genuine changes in the scene geometry.

We describe a simple generative probabilistic model for 4D structure in which surface patches, specified by a spatial location, orientation and temporal extent, generate point and normal observations over time. These observations are only recorded by the sensor if they fall within the spatio-temporal field of view of a measurement and are not occluded by other surfaces patches which exist at that same time. We assume the scene is static and rigid for the duration of each measurement and leave the problem of spatio-temporal grouping of surface patches into object tracks

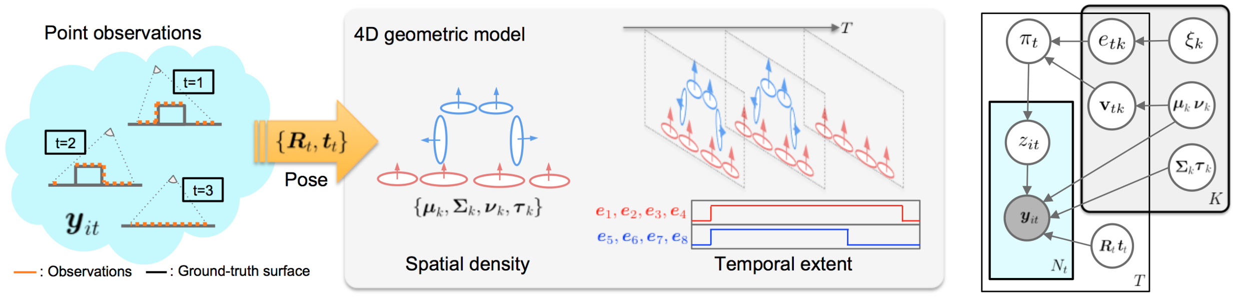


Figure 5.2: Observations are explained by a collection of surface patches that exist for some temporal extent and emit point observations into some unknown local coordinate system. We use a probabilistic mixture model (right) that represents parameters for K patches producing N_t point observations at each of T time points. The prior probability of a patch emitting an observation at a given time $\boldsymbol{\pi}_t$ depends on whether the patch exists $\mathbf{e}_{tk}$ and is visible $\mathbf{v}_{tk}$, which in turn depends on the space-time geometry of the scene $(\boldsymbol{\xi}_k, \boldsymbol{\mu}_k, \boldsymbol{\nu}_k)$.

across time points as a post-process. Fitting the model to a time-stamped stream of measurements yields an estimate of the 4D scene structure as well as the location of the camera at each measurement time point. We term this problem *Space-Time SLAM* since it generalizes the standard offline RGB-D SLAM problem with a 4th temporal dimension[1].

The chief merits of this approach are in (1) providing robust pose estimation in the presence of changing scenes and (2) producing scene reconstructions that more accurately reflect the scene geometry at any given time point. In particular, by reasoning about occlusion, the model is capable of inferring amodal completion of surfaces which may be hidden by an object during some times but later revealed when the object moves. We quantify these benefits relative to baselines that lack an explicit temporal model using a synthetic dataset where ground-truth geometry and pose are known. We also perform comparisons using challenging data collected from several indoor workspaces over the course of several weeks. Finally, we demonstrate the utility of the recovered 4D model in segmenting dynamic objects by simply clustering surface patches based on their temporal extent.

[1]We assume the temporal coordinate of each measurement is given while a full generalization of SLAM would also estimate the 6+1 DOF camera space-time pose

5.1 Space-time model fitting

We now describe our model for 4D geometry as a collection of surface patches with specified location, spatial and temporal extents. The model is fit to 3D point observations using a generative probabilistic approach inspired by the joint registration methods of Horaud *et al.* [44] and Georgios *et al.* [22]. For model overview, please refer to Figure 5.2.

5.1.1 Notation and Model Formulation

Surface patches We model the scene as a collection of K surface patches. Each patch has a mean parameter $(\boldsymbol{\mu}_k, \boldsymbol{\nu}_k) \in \mathbb{R}^3 \times \mathbb{S}^2$ that describes the location and orientation. The spatial extent and roughness of the surface patch is described by corresponding variance parameters $(\boldsymbol{\Sigma}_k, \tau_k)$. Additionally, each patch k has a specific temporal extent during which it exists in the scene specified by a time interval $[a_k, b_k]$. We denote the collection of shape parameters by $\mathcal{X} = \{(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\nu}_1, \tau_1), \ldots\}$ and temporal parameters by $\xi = \{(a_1, b_1), (a_2, b_2), \ldots\}$. We use the binary vector $\mathbf{e}_{tk} \in \{0, 1\}$ to indicate if patch k *exists* at time t so that $\mathbf{e}_{tk} = 1$ iff $t \in [a_k, b_k]$.

Observations The input data stream consists of observations of scene structure $\mathcal{Y} = \{\mathbf{y}_1 \ldots \mathbf{y}_T\}$ at T discrete times. The observation at time t consists of N_t points with surface normals $\mathbf{y}_t = \{(\boldsymbol{l}_{it}, \boldsymbol{n}_{it}) \in \mathbb{R}^3 \times \mathbb{S}^2\}_{1 \leq i \leq N_t}$ where $\boldsymbol{l}_{it}$ and $\boldsymbol{n}_{it}$ to denote the location and surface normal associated with observation $\boldsymbol{y}_{it}$. In our experiments this data comes from a scan acquired by an RGB-D sensor but could come from other sources (e.g., ToF laser scanner or SfM reconstruction).

Pose and Occlusion Individual observations are assumed to be metric but are recorded in an arbitrary local coordinate system specified by unknown pose parameters $\Theta = \{\mathbf{R}_t, \mathbf{t}_t\}_{1 \leq t \leq T}$ which vary across time. We estimate a rigid transformation ϕ_t mapping each observation into a single global coordinate system. $\phi_t(\boldsymbol{l}_{it}, \boldsymbol{n}_{it}) = (\mathbf{R}_t\boldsymbol{l}_{it} + \mathbf{t}_t, \mathbf{R}_t\boldsymbol{n}_{it})$. A patch k may not be visible at time t due to the sensor placement relative to the scene structure. We use the variable $\mathbf{v}_{tk} \in \{0, 1\}$ indicate whether patch k is visible at time t which depends on the 4D model $\mathcal{X}$ and sensor parameters.

Generating Observations from Patches To generate observed data at time t from scene $(\mathcal{X}, \xi)$ with a specified camera placement, we select a surface patch k at random from those patches present at time t. If the patch is visible from the sensor position, then we sample a point location and normal from the patch density with parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$. To allow for noise in the observations, we also include a background noise component whose distribution is uniform over the volume of a bounding box enclosing the scene model.

For a given time t, the probability of generating an observation $\boldsymbol{y}$ in local coordinates is modeled as a probabilistic mixture:

$$P(\boldsymbol{y}_t|\mathcal{X}, \Theta, \boldsymbol{\xi}) = \sum_{k=0}^{K} P(\phi_t(\boldsymbol{y}_t)|\mathcal{X}_k)P(k|\mathbf{e}_t, \mathbf{v}_t) \tag{5.1}$$

The probability of generating an observation from patch k is given by:

$$\pi_t(k) = P(k|\mathbf{e}_t, \mathbf{v}_t) = \begin{cases} \frac{1}{Z_t}p_0 & k = 0 \\ \frac{1}{Z_t}p_k e_{tk}\mathbf{v}_{tk} & k \in \{1 \ldots K\} \end{cases}$$

where p_k is the time-independent intensity with which a patch k generates observations, p_0 is the background noise intensity, and Z_t is a normalizing constant.

Observations associated with a given patch are transformed into the global coordinate frame with location modeled by a Gaussian density and unit surface normal modeled by a von Mises Fisher (vMF) density

$$\begin{aligned} P(\phi_t(\boldsymbol{l},\boldsymbol{n})|\mathcal{X}_k) &= P(\phi_t(\boldsymbol{l})|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)P(\phi_t(\boldsymbol{n})|\boldsymbol{\nu}_k,\tau_k) \\ P(\phi_t(\boldsymbol{l})|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) &= \frac{1}{Z(\boldsymbol{\Sigma}_k)}\exp(-\frac{1}{2}||(\boldsymbol{R}_t\boldsymbol{l}+\boldsymbol{t}_t)-\boldsymbol{\mu}_k||^2_{\boldsymbol{\Sigma}_k}) \\ P(\phi_t(\boldsymbol{n})|\boldsymbol{\nu}_k,\tau_k) &= \frac{1}{Z(\tau_t)}\exp(\tau_k\boldsymbol{n}^T\boldsymbol{R}_t^T\boldsymbol{\nu}_k) \end{aligned}$$

Background noise points ($k = 0$) are drawn from a uniform distribution over the observation volume $\boldsymbol{y} \sim \mathcal{U}(V \times \mathbb{S}^2)$.

5.1.2 Computing Visibility

To assign observations to surface patches, we need to compute visibility variables $\mathbf{v}_{tk}$ that indicate if patch k is visible at time t. In order for a patch to be observed it must fall within the field of view of the sensor and must not be occluded by any other surface that existed at the observation time. For each time, we have one or more sets of camera parameters associated with gathering observations $\mathbf{y}_t$ which we write as $\{\mathbf{C}_{tu}\}_{1\leq u\leq F_t}$ where F_t is the number of RGB-D frames used to build the observation.

Let ϕ_t^{-1} be the transformation specified by parameters Θ_t that transforms the global model $\mathcal{X}$ into the local coordinate frame used by observations at time t. We define the indicator function $FOV(\phi_t^{-1}(\boldsymbol{\mu}_k,\boldsymbol{\nu}_k),\mathbf{C}_{tu})$ to be 1 if patch k was in the field of view of camera $\mathbf{C}_{tu}$.

To estimate occlusion, we use the hidden point removal algorithm introduced in [94] applied to the union of the observed points $\mathbf{y}_t$ and the set of transformed patch locations $\phi_t^{-1}(\mathcal{X})$ which are in the field of view. Let $OCL(\phi_t^{-1}(\boldsymbol{\mu}_k),\mathbf{y}_t,\mathbf{C}_{tu})$ be 1 if $\phi_t^{-1}(\boldsymbol{\mu}_k)$ is occluded by some part of $\mathbf{y}_t$ from camera viewpoint $\mathbf{C}_{tu}$ and 0 otherwise.

Combining these two components, we estimate that a patch k should be visible at time t if it is within the field of view and unoccluded in at least one camera view used to construct observation $\mathbf{y}_t$.

$$\mathbf{v}_{tk} = \begin{cases} 1 & \begin{array}{l} \exists u : [FOV(\phi_t^{-1}(\boldsymbol{\mu}_k, \boldsymbol{\nu}_k), \mathbf{C}_{tu}) = 1] \wedge \\ \quad [OCL(\phi_t^{-1}(\boldsymbol{\mu}_k), \mathbf{y}_t, \mathbf{C}_{tu}) = 0] \end{array} \\ 0 & \text{otherwise} \end{cases}$$

5.1.3 Model Parameter Estimation

To fit the model, we maximize the log-likelihood of observing $\mathcal{Y}$ given transformation parameters Θ and space-time geometry $\{\mathcal{X}, \xi\}$ assuming independent point observations:

$$\max_{\mathcal{X},\Theta,\boldsymbol{\xi}} \log \prod_{it} P(\boldsymbol{y}_{it}|\mathcal{X}, \Theta, \boldsymbol{\xi}) P(\mathcal{X}, \Theta, \boldsymbol{\xi})$$

We assume an uninformative priors on $\mathcal{X}$ and Θ and a prior on $\boldsymbol{\xi}$ that favors longer intervals (see below for details).

Let z_{it} be a latent variable that denotes mixture assignments with $z_{it} = k$ when $\boldsymbol{y}_{it}$ is a point from patch k. We use expectation conditional maximization (ECM), which alternates between estimating expectations of $\mathcal{Z}$ and conditionally optimizing subsets of model parameters [79]. For a fixed setting of model parameters, ($\mathcal{X}$,Θ,$\boldsymbol{\xi}$), the *E-step* estimates the probability that each observation y_{it} came from surface patch k.

$$\alpha_{itk} \quad = \quad P(z_{it} = k \mid \mathcal{X}, \Theta, \boldsymbol{\xi})$$

During the *M-step* we maximize the expected likelihood of subsets of parameters sequentially conditioned on the other parameters. Letting s denote the iteration, we first update the alignment parameters, followed by the scene geometry and finally the temporal extent.

$$\begin{aligned}
\Theta^{s+1} &= \arg\max_{\Theta} E_{\mathcal{Z}}\big[\log\big(P(\mathcal{Y},\mathcal{Z};\mathcal{X}^s,\Theta,\boldsymbol{\xi}^s)\big)\big] \\
\mathcal{X}^{s+1} &= \arg\max_{\mathcal{X}} E_{\mathcal{Z}}\big[\log\big(P(\mathcal{Y},\mathcal{Z};\mathcal{X},\Theta^{s+1},\boldsymbol{\xi}^s)\big)\big] \\
\boldsymbol{\xi}^{s+1} &= \arg\max_{\boldsymbol{\xi}} E_{\mathcal{Z}}\big[\log\big(P(\mathcal{Y},\mathcal{Z};\mathcal{X}^{s+1},\Theta^{s+1},\boldsymbol{\xi})\big)\big]
\end{aligned}$$

We provide details for each parameter update below.

Patch Assignment Given the aligning transformation $\phi_t(\cdot)$ for time t along with geometric, existence and visibility terms for K patches, we compute the posterior probability that an observation is generated by a particular patch as:

$$\alpha_{itk} = \frac{P(\phi_t(\boldsymbol{y}_{ti})|\mathcal{X}_k)\pi_t(k)}{\sum_{j=1}^{K} P(\phi_t(\boldsymbol{y}_{ti})|\mathcal{X}_j)\pi_t(k)+\beta}$$

where $\pi_t(k)$ is the mixing weight of a patch k at time t, β is the weight of the background/outlier cluster which depends on p_0, and we set $\alpha_{it0} \propto \beta$ (see [44] for details).

Alignment Parameters Given the cluster assignment expectations for each observation, we would like to update the estimated transformation parameters $\mathbf{R}_t$ and $\mathbf{t}_t$ for t-th dataset. This amounts to a weighted least-squares problem with orthogonality constraint on $\mathbf{R}_t$. Following [44], we simplify this expression by first constructing a single "virtual point" $\mathbf{u}_{tk}$ per mixture component that integrates the interaction of all observed points with the patch.

$$\mathbf{u}_{tk} = \mathbf{w}_{tk} \sum_{i=1}^{N_t} \alpha_{itk} \boldsymbol{l}_{ti} \quad \mathbf{w}_{tk} = \left(\sum_{i=1}^{N_t} \alpha_{itk} \right)^{-1}$$

The optimal transformation can then be expressed as a weighted rigid alignment problem:

$$\arg\min_{\mathbf{R},\mathbf{t}} \frac{1}{2} \sum_{k=1}^{\mathbf{K}} \mathbf{w}_{tk} ||\mathbf{R}\mathbf{u}_k + \mathbf{t} - \boldsymbol{\mu}_k||^2_{\boldsymbol{\Sigma}_k}$$

When $\boldsymbol{\Sigma}_k$ is isotropic this can be solved efficiently using SVD (e.g., [46]). This provides a good initialization that can be further refined by projected gradient for anisotropic case and the additional linear term from the density over surface normals.

Spatial Patch Parameters Given the transformation parameters, we update mean and covariance for each Gaussian mixture component.

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{\sum_{t=1}^{T} \sum_{i=1}^{N_t} \alpha_{itk} \phi_t(\boldsymbol{l}_{it})}{\sum_{t=1}^{T} \sum_{i=1}^{N_t} \alpha_{itk}} \\
\boldsymbol{\Sigma}_k &= \frac{\sum_{t=1}^{T} \sum_{i=1}^{N_t} \alpha_{itk} \left(\phi_t(\boldsymbol{l}_{it}) - \boldsymbol{\mu}_k\right) \left(\phi_t(\boldsymbol{l}_{it}) - \boldsymbol{\mu}_k\right)^{\mathsf{T}}}{\sum_{t=1}^{T} \sum_{i=1}^{N_t} \alpha_{itk}} + \epsilon^v \mathbf{I}
\end{aligned}$$

The variable ϵ^v is used to prevent the variance of a given cluster from collapsing ($\epsilon^v = 0.1^6$ in our experiments). The updates for vMF mean and concentration parameters for the surface normal follow a similar form [36]. In practice, we found that constraining the covariance to be isotropic works well for large K and yields more efficient optimization. Patch intensity priors p_k are estimated as the assignment proportion $\sum_{it} \alpha_{itk}$ scaled by the proportion of observations in which the patch was visible and existed.

Temporal Patch Parameters To reliably estimate when each patch is present, we must make some stronger assumptions about the prior distribution over $\mathbf{e}_k$. We thus assume that a given patch exists for a single temporal interval $[a_k, b_k]$ during which the probability of the patch emitting an observation is uniform.

$$P(e_{tk} = 1) = \begin{cases} \gamma_{ab} & t \in [a_k, b_k] \\ \epsilon & \text{otherwise} \end{cases}$$

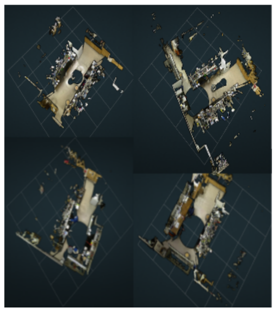
where ϵ is a small constant and γ is chosen so the distribution integrates to 1 over the total observation interval T.

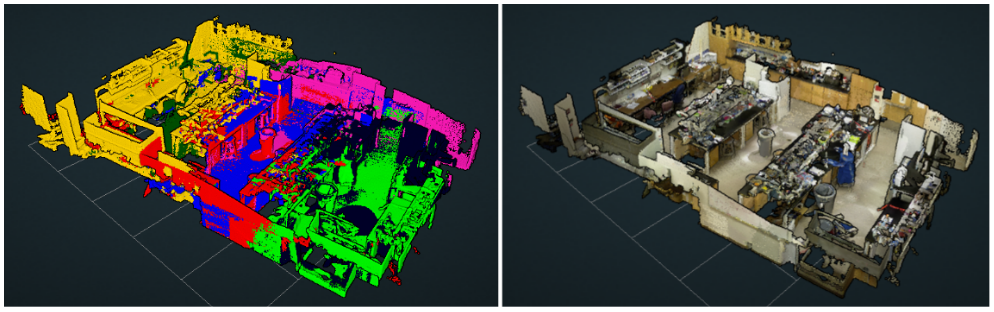
$$\gamma_{ab} = \frac{1 - \epsilon(T - (b - a))}{(b - a)},$$

To estimate a_k, b_k we maximize the expected posterior probability over times where the cluster was visible:

$$[a_k, b_k] = \arg\max_{a,b} \sum_{t \in [a,b]} \mathbf{v}_{tk} \left[\sum_{i=1}^{N_t} \alpha_{itk} \log(\gamma_{ab}) \right] + \sum_{t \notin [a,b]} \mathbf{v}_{tk} \left[\sum_{i}^{N_t} \alpha_{itk} \log(\epsilon) \right] + \log P^s(a, b)$$

where $\log P^s(a, b) = avg(\alpha_{\cdot\cdot k}) \log(b - a + \epsilon^p)$ is a prior that encourages existence of patches for longer time spans. The prior is scaled using average value of $\alpha_{\cdot\cdot k}$. In our experiments we use 0.05 for ϵ and 0.01 for ϵ^p.


(a) Observations


(b) Aligned observations from the same time point

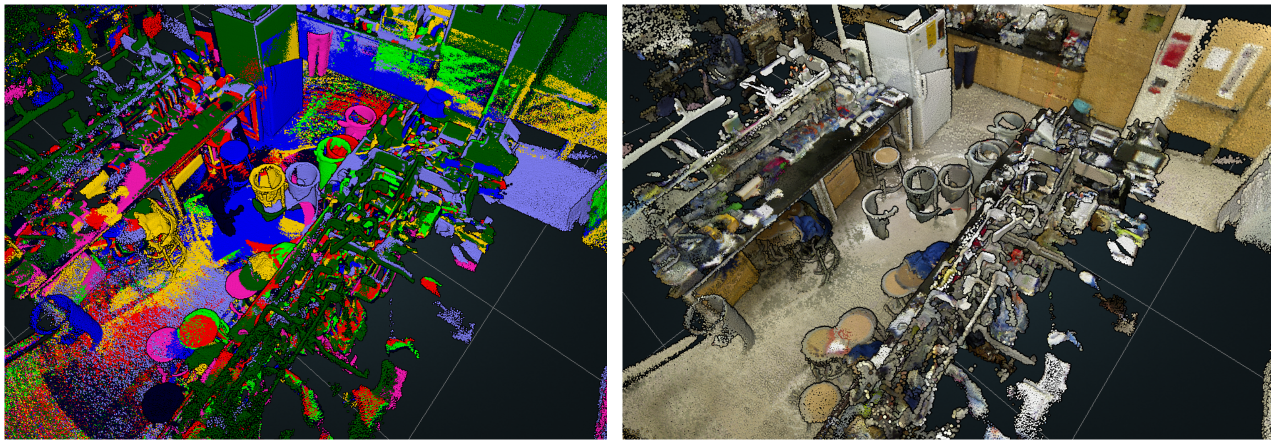

(c) Merged observations from all time points. Note that per each time point, there was only one trash bin.

Figure 5.3: 3D model pieces with 4 different types (a) for "Laboratory" dataset. (b) and (c) shows merged model with one time or space.

Extension to Non-parametric Mixtures In addition to standard mixture model fitting with fixed number of clusters K, we also considered a variant of our model using a Dirichlet Process (DP) prior over the cluster allocations [104, 69, 68]. This is appealing since it allows the model to naturally grow in complexity as more observations become available. We use collapsed Gibbs sampling to explore the space of the number of mixture components K and weights π. Rather than performing full Bayesian inference, we interleaved rounds of sampling with conditional maximization to optimize alignment parameters. We observed empirically that starting from an initial state with few mixture components and refining the alignment while non-parametrically growing the number of components often resulted in better registration results (see experiments). We presume this may be because the early energy landscape with few mixture components has fewer local minima.

5.2 Space-Time Datasets

While there are a large number of published RGB-D and 3D scene datasets (e.g., [119]), previous work has focused on static scenes described at a single point in time (or collected at high-frame rate over a short interval). To validate our approach with more compelling temporally varying elements, we developed datasets based on both synthetic scenes with simulated sensors and real scans of human workspaces

Synthetic Data Synthetic data is easy to generate and provides perfect ground-truth which is useful for evaluating reconstruction accuracy. To emulate the noise characteristics of real scanning, we start from a 3D model and simulate acquisition of RGB-D data from a moving sensor and pass it through a standard SLAM pipeline to produce a 3D point cloud which constitutes observation at a single time point.

We use 3D room models provided by [40] and populate them with IKEA furniture models. Each item of furniture is present for randomly specified interval of time. We generate a virtual sensor trajectory by selecting several key points manually and synthesize a smooth path connecting the key points. Given a trajectory we render a sequence of RGB-D frames which are then fed into the ElasticFusion [116] pipeline to produce a simulated observed point-cloud. Back-projecting key-points from the observation provides a ground-truth alignment with the world coordinate system and mapping between the simulated observed points and the object ids in the scene model. We generate 8 time point per scene producing scans with a million points (summarized in Table 5.1).

Real Data We also collected scans of 3 different indoor scenes (Laboratory, Kitchen, and Copier room) once a day over several weeks using a Kinect sensor. We chose these scenes since they contained a number of objects that naturally move from one day to the next by people passing through the room. To provide the best quality and consistency in scanning, we used a custom

Synthetic Data

Name	# regions	# times	# frames	# 3D points
Bedroom	1	8	3.2k	1.2M
Bathroom	1	8	4k	1.5M

Real Data

Name	# regions	# times	# frames	# 3D points
Laboratory	4	8	3k	1M
Copier room	1	7	1.5k	0.6M
Kitchen	1	5	1.5k	0.6M

Table 5.1: Summary statistics of test datasets.

motorized tripod fixture to automated the scan path. For our "Laboratory" dataset, we collected 4 scans per time point in order to cover different overlapping parts of a larger room (see Figure 5.3). Each scan is processed individually using ElasticFusion [116] to produce a point cloud. Dataset statistics are summarized in Table 5.1.

To establish a high quality ground-truth alignment, we exploit the presence of the floor which is visible in all our recovered scans. We first segment the floor based on color and surface normal from each scan. We then constrain the search over alignments to only consider translations in the plane of the floor and rotations around the z axis perpundicular to the floor. This pre-process greatly reduces the number local minima and, guided by a few hand-clicked correspondences, is sufficient for finding high quality alignments which can be further refined to improve accuracy.

Once the scans are aligned into a common coordinate frame, we segment and annotate the points in each scan with object instance labels such as "floor", "desk", "chair", etc. These instance labels are shared across time points, allowing us to identify observations at different times which correspond to the same underlying surface and provide a basis for benchmarking the ability of our model to correctly identify spatio-temporal extents (see Figure 5.6).

5.3 Experimental Evaluation

Figure 5.3 shows qualitative results of running our joint registration and reconstruction model on the Laboratory dataset depicting (a) individual scans, (b) reconstruction of a single time point consisting of multiple overlapping scans, and (c) all time point reconstructions superimposed in a single global coordinate system. Colors (b) and (c) indicate points belonging to different scans.

We quantitatively evaluate the method in terms of the accuracy of reconstruction and localization (alignment). In particular, we show that the existence and visibility terms are valuable even when aligning partially overlapping scans for static scenes. We then evaluate the accuracy of estimated existence time intervals. Lastly, we demonstrate the utility of this 4D representation in segmenting out dynamic objects in a scene.

Spatial reconstruction accuracy To evaluate metric reconstruction quality, we compare our method to two baselines which don't model temporal change. First, we consider running the ElasticFusion pipeline applied to data concatenated from all time points (**EF3D**). Second, we consider running ElasticFusion independently at each time point and subsequently align reconstructions from different times using the method of Evangelidis *et al.* [22] (**EF4D**). EF3D produces a single 3D reconstruction which is compared to all time points while EF4D and our model produce 4D reconstructions which change over time. We evaluate precision and recall (of those model points that were visible from the sensor) w.r.t. ground-truth surface across all time points for the synthetic 4D benchmark using a distance threshold of 1cm.

As Table 5.2 shows, the precision of a single 3D reconstruction (EF3D) is lower than the 4D models (EF4D,ours). Our model further improves precision over simply aligning individual time points by providing more robustness to dynamic objects. Our method also shows a substantial boost in recall over both baselines due to the ability of the model to fill in occluded regions with observations from a different time.

Temporal reconstruction accuracy We use the synthetic dataset for which the ground-truth duration of existence of each object is known and evaluate the accuracy (Table 5.3). Since our model predicts existence of patches, we establish a correspondence, assigning points of each

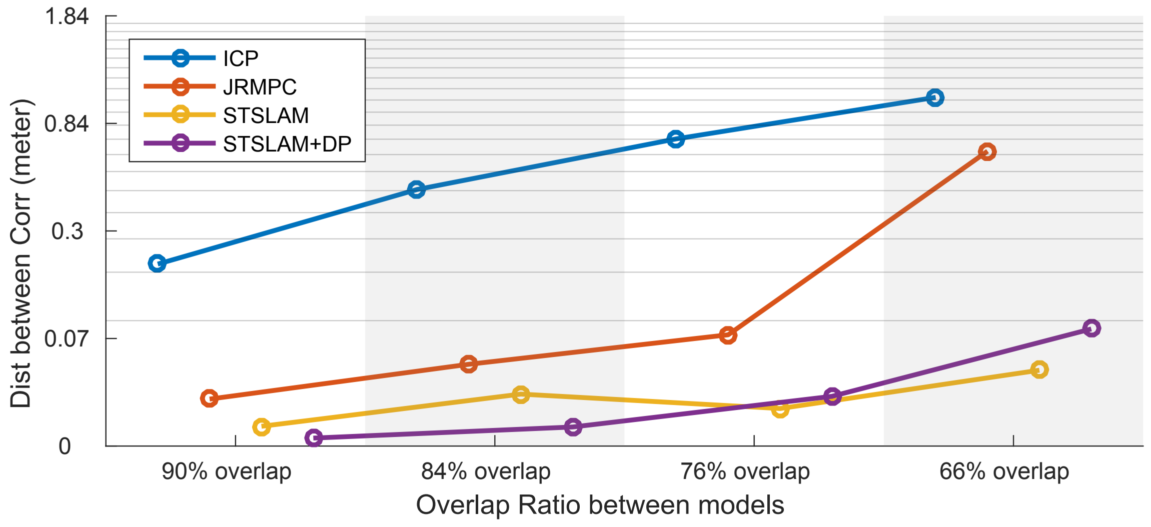


Figure 5.4: Registration accuracy of our method (ST-SLAM) compared to (ICP) [8] and joint registration (JRMPC) [22] measured by average closest point distance. Error axis is on a logarithmic scale. Explicitly inferring visibility improves robustness to partial overlap, as does using non-parametricly growing the number of mixture components during optimization (ST-SLAM+DP). Error in estimated rigid transformation parameters behaves similarly (see supplement).

ground-truth object to a mixture component in our estimated model. This assignment imposes an upper-bound on the accuracy (i.e., since a mixture component may span two different objects whose temporal extent differs). We find that most incorrect predictions come from near such edges.

Density visualization Since our reconstruction is generative, we can also visualize it as a spatio-temporal probability density. In Figure 5.5, we display the estimated density of observations marginalized over all time as well as conditioned on specific time points alongside the corresponding scene. To aid visualization, we exclude the background scene component and rescale the colormap. As the figure shows, the estimated space-time density tracks the arrangement of furniture within the room.

Robustness on partially overlapping observations Previous joint registration methods [8, 54, 22] that align multiple pointclouds into a single consensus model often rely on a high degree of overlap between scans in order to achieve correct alignment. Since our approach explicitly models

	EF3D	EF4D	Our method (STSLAM)
Precision	71.3%	92.7%	93.3%
Recall	90.4%	90.0%	98.5%

Table 5.2: Evaluation for reconstruction quality. Baselines EF3D merges all observations in a single 3D model; EF4D builds a separate model for each time point and then registers them. Our model of temporal extent and visibility improves both precision and recall. See Section 5.3 for details.

	Baseline	STSLAM	STSLAM-DP	Upper-bound
Bedroom	83.2%	90.8%	90.9%	95.5%
Bedroom (NS)	41.2%	74.3%	74.8%	95.4%
Bathroom	69.7%	78.0%	77.7%	95.0%
Bathroom (NS)	31.8%	62.2%	62.7%	99.9%

Table 5.3: Temporal reconstruction accuracy. Baseline assumes every object exists for all time points. Non-static (NS) evaluation excludes the static background component. Upper-bound indicates maximal achievable accuracy given the spatial quantization imposed by cluster assignment.

which surface patches are visible in a given scan, it can handle larger non-overlapping regions by allocating additional mixture components and explaining away the lack of data generated from those components in scans where they were not visible.

To demonstrate the value of estimating visibility, we carry out an experiment on the ground-truth Laboratory reconstruction by splitting a single time point into two pieces with controlled degree of overlap, ranging from 50% to 90%, apply a random rigid transformation and add Gaussian noise to point locations. We measure difference between the known transformation and the estimate, as well as mean distance between corresponding points, averaged over 10 trials.

As Figure 5.4 displays, all methods have higher registration error as the degree of overlap decreases. Since ICP [8] and the joint registration method JRMPC [22] do not infer which consensus points/mixtures are visible in a given observation, their performance degrades more rapidly as overlap decreases. Our model with a fixed number of mixtures (ST-SLAM) is more robust and using the DP mixture allocation (ST-SLAM+DP) yields more robust results, presumably due to annealing effects of starting with a small number of mixture components.

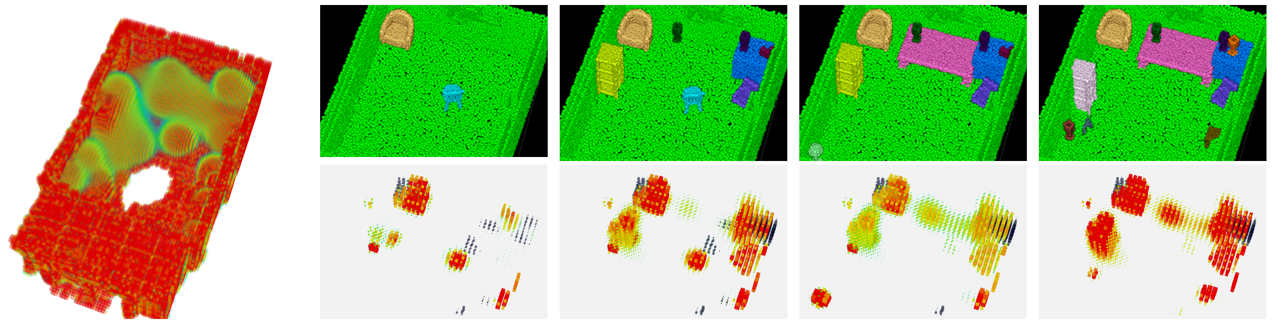

(a) Marginal density $P(\mathbf{y})$ (b) $P(\mathbf{y}|t=1)$ (c) $P(\mathbf{y}|t=2)$ (d) $P(\mathbf{y}|t=3)$ (d) $P(\mathbf{y}|t=4)$

Figure 5.5: Visualization of 3D probability density predicted by the fit model. (a) shows density marginalized over time (b) - (e) display probability density conditioned on different observation times with the static component excluded for clarity.

Segmentation of dynamic objects The space-time geometry model provides a natural basis for performing additional inferences about a scene. In particular, the temporal coherence of a set of surface patches provides a strong indicator that those patches belong to the same surface. In Figure 5.6 we visualize segmentations into objects based on grouping those patches with a similar estimated temporal extent. Raw measurements are assigned the segment label for that cluster with the highest assignment probability (α_{itk}). As shown in the upper figure panel, the segmentation accuracy is limited based on the size/number of surface patches fit to the scene. To produce good quality assignments we choose a number of mixture components that yields patches of an average small physical dimension (relative to the resolution of the raw point observations). The lower panel of Figure 5.6 shows such a segmentation with the static background component in green.

We consider two quantitative measures of segmentation accuracy. Let U, V, S denote the surface patches, predicted segments, and ground-truth segments respectively. We characterize the degree of under-segmentation (i.e., how often a surface patch spans an object boundary) by the average percentage of a patch that is completely contained in some ground-truth segment.

$$Score_{useg} = \frac{1}{N_c}\sum_{i=1}^{N_c}\max_j \frac{|U_i \cap S_j|}{|S_j|}.$$

To measure the effectiveness of grouping patches by temporal extent, we compute the IoU of predicted and ground-truth segments.

$$Score_{seg} \quad = \quad \frac{1}{N_s}\sum_{i=1}^{N_s}\max_j \frac{|V_i \cap S_j|}{|V_i \cup S_j|}$$

In Figure 5.7 we plot these scores as a function of the number of mixture components. As might be expected, the under-segmentation error decreases rapidly as the number of clusters grow, allowing smaller patches that are less likely to span an object boundary. However, there is a tradeoff in segmentation accuracy of dynamic objects as the number of clusters goes beyond a certain point as the estimates of temporal extent become increasingly noisy with few observations per cluster.

5.4 Discussion

In this chapter, we have proposed a novel probabilistic formulation of space-time localization and mapping from RGB-D data streams which jointly estimates sensor pose and builds an explicit 4D map. We validated this approach on real and synthetic data, showing improved reconstruction and registration for dynamic scenes and demonstrate unique features of the model which allow estimation of the temporal extent of surface patches and segmentation into temporally coherent objects. In the future we hope to extend this approach to handle more dynamic and larger-scale scenes, replace our wide-FoV observations with individual RGB-D frames, and tackle the problem of inter-frame tracking of moving objects.

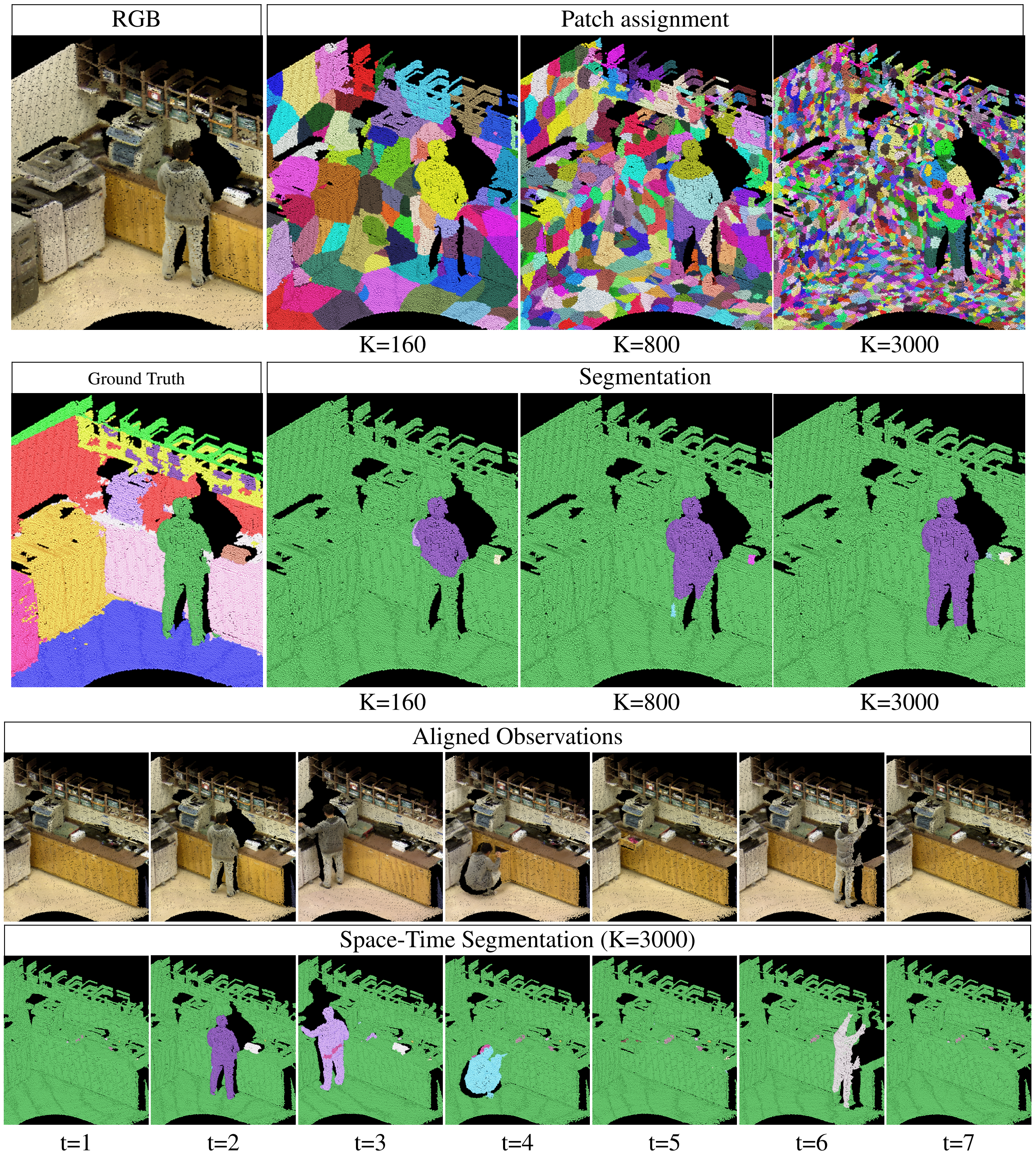
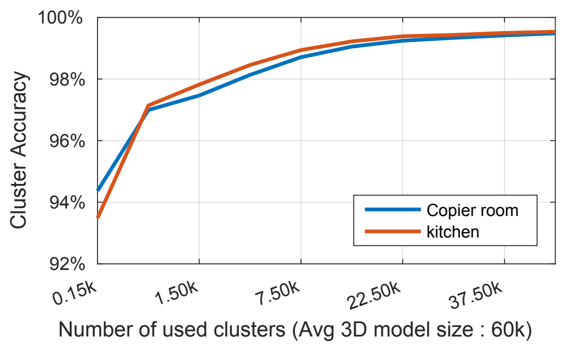
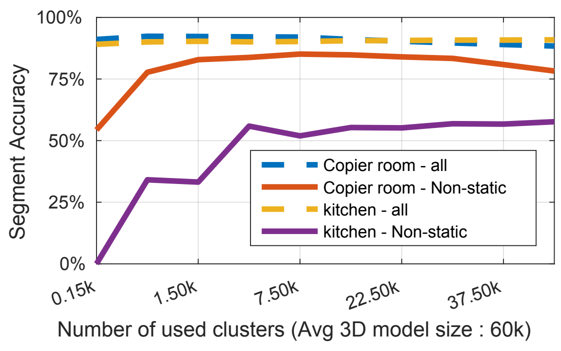


Figure 5.6: Segmentation results on the “Copier room” dataset showing grouping of surface patches with similar temporal extent. Segmentation accuracy depends on the number of surface patches (top). Segmentation across all space-time observations using the optimal cluster size discovers static and dynamic scene components (bottom).



(Under-segmentation accuracy)



(Segmentation accuracy)

Figure 5.7: We measure the accuracy with which individual patches constitute an under-segmentation of the objects in the scene (left) and how well grouping patches by temporal extent recovers object segments (right) on real datasets "Copier room" and "Kitchen". Since the scene is dominated by static structure, we also separately plot the segmentation accuracy for the non-static components.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building Rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[3] J. Aggarwal, B. Vemuri, Y. Chen, and G. Medioni. Range image understanding object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145 – 155, 1992.

[4] Y. Almalioglu, M. R. U. Saputra, P. P. de Gusmao, A. Markham, and N. Trigoni. Ganvo: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[5] R. Ambrus, V. Guizilini, J. Li, S. Pillai, and A. Gaidon. Two stream networks for self-supervised ego-motion estimation, 2019.

[6] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *IEEE CVPR*, pages 1534–1543, 2016.

[7] V. Babu, A. Majumder, K. Das, S. Kumar, et al. A deeper insight into the undemon: Unsupervised deep network for depth and ego-motion estimation. *arXiv*, 2018.

[8] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE TPAMI*, 14(2):239–256, Feb. 1992.

[9] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *IEEE CVPR*, volume 2, pages 690–696, 2000.

[10] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625, 2012.

[11] 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*, 32(6):1309–1332, 2016.

[12] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8001–8008, 2019.

[13] Y. Chen, C. Schmid, and C. Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. *arXiv preprint arXiv:1907.05820*, 2019.

[14] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *IEEE CVPR*, 2015.

[15] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998.

[16] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. Are we ready for autonomous drone racing? the uzhfpv drone racing dataset. In *IEEE Int. Conf. Robot. Autom.(ICRA)*, 2019.

[17] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[18] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[19] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 38(2):295–307, 2016.

[20] A. Dosovitskiy, P. Fischery, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015.

[21] J. Engel and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.

[22] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis. A generative model for the joint registration of multiple point sets. In *ECCV*, 2014.

[23] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, et al. Building Rome on a cloudless day. In *ECCV*, pages 368–381, 2010.

[24] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE TPAMI*, 2010.

[25] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[26] D. Gálvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[27] R. Garg, V. K. B.G., G. Carneiro, and I. Reid. *Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue*, pages 740–756. 2016.

[28] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[29] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[30] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas, and N. Navab. Optical flow estimation with uncertainties through dynamic mrfs. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[31] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[32] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[33] C. Godard, O. Mac Aodha, and G. J. Brostow. Digging into self-supervised monocular depth estimation. *ICCV*, 2019.

[34] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese. Monitoring changes of 3d building elements from unordered photo collections. In *IEEE ICCV Workshops*, pages 249–256, 2011.

[35] V. Golyanik, B. Taetz, G. Reis, and D. Stricker. Extended coherent point drift algorithm with correspondence priors and optimal subsampling. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, 2016.

[36] S. Gopal and Y. Yang. Von Mises-Fisher clustering models. In *ICML*, pages 154–162, 2014.

[37] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. *CoRR*, 2019.

[38] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[39] A. Handa, M. Bloesch, V. Pătrăucean, S. Stent, J. McCormac, and A. Davison. gvnn: Neural network library for geometric computer vision. In *ECCV*, pages 67–82. Springer, 2016.

[40] A. Handa, V. Patraucean, S. Stent, and R. Cipolla. Scenenet: An annotated model generator for indoor scene understanding. In *IEEE International Conference on Robotics and Automation, (ICRA)*, 2016.

[41] D. J. Heeger and A. D. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992.

[42] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.

[43] A. Hermans, G. Floros, and B. Leibe. Dense 3d semantic mapping of indoor scenes from RGB-D images. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 2631–2638, 2014.

[44] R. Horaud, M. Yguel, G. Dewaele, F. Forbes, and J. Zhang. Rigid and articulated point registration with expectation conditional maximization. *IEEE TPAMI*, 33:587–602, 2010.

[45] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[46] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, 1987.

[47] Z. Huang, C. Wan, T. Probst, and L. V. Gool. Deep learning on lie groups for skeleton-based action recognition. *CVPR*, pages 1243–1252, 2017.

[48] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *CVPR*, 2016.

[49] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[50] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu. Spatial transformer networks. In *NIPS*, pages 2017–2025. Curran Associates, Inc., 2015.

[51] A. Jaegle, S. Phillips, and K. Daniilidis. Fast, robust, continuous monocular egomotion computation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 773–780, 2016.

[52] A. Jaegle, S. Phillips, and K. Daniilidis. Fast, robust, continuous monocular egomotion computation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 773–780. IEEE, 2016.

[53] M. Janner, J. Wu, T. Kulkarni, I. Yildirim, and J. B. Tenenbaum. Self-Supervised Intrinsic Image Decomposition. In *NIPS*, 2017.

[54] B. Jian and B. Vemuri. Robust point set registration using Gaussian mixture models. *IEEE TPAMI*, 33(8):1633–1645, 2011.

[55] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[56] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *IROS*, 2013.

[57] D. H. Kim and J. H. Kim. Effective background model-based rgb-d dense visual odometry in a dynamic environment. *IEEE Transactions on Robotics*, 32(6):1565–1573, 2016.

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

[59] B. Klingner, D. Martin, and J. Roseborough. Street view motion-from-structure-from-motion. In *IEEE ICCV*, pages 953–960, 2013.

[60] 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)*, pages 698–713, 2018.

[61] 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*, pages 3607–3613. IEEE, 2011.

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

[63] M. Lee and C. C. Fowlkes. Cemnet: Self-supervised learning for accurate continuous ego-motion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[64] R. Li, S. Wang, Z. Long, and D. Gu. UnDeepVO: Monocular visual odometry through unsupervised deep learning. *arXiv*, 2017.

[65] R. Li, S. Wang, Z. Long, and D. Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291. IEEE, 2018.

[66] S. Li and D. Lee. Rgb-d slam in dynamic environments using static point weighting. *IEEE Robotics and Automation Letters*, 2(4):2263–2270, 2017.

[67] Y. Li, Y. Ushiku, and T. Harada. Pose graph optimization for unsupervised monocular visual odometry. 2019.

[68] D. Lin. Online learning of nonparametric mixture models via sequential variational approximation. In *NIPS*, pages 395–403. 2013.

[69] D. Lin and J. W. Fisher. Coupling nonparametric mixtures via latent dirichlet processes. In *NIPS*, pages 55–63. 2012.

[70] P. Liu, M. Lyu, I. King, and J. Xu. Selflow: Self-supervised learning of optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[71] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[72] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *arXiv preprint arXiv:1810.06125*, 2018.

[73] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, June 2018.

[74] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018.

[75] R. Martin-Brualla, D. Gallup, and S. M. Seitz. 3d time-lapse reconstruction from internet photos. In *IEEE ICCV*, 2015.

[76] K. Matzen and N. Snavely. Scene chronology. In *ECCV*, pages 615–630, 2014.

[77] S. Meister, J. Hur, and S. Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[78] I. Melekhov, J. Kannala, and E. Rahtu. Relative camera pose estimation using convolutional neural networks. *arXiv*, 2017.

[79] X.-L. Meng and D. B. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267, 1993.

[80] N. J. Mitra, S. Flöry, M. Ovsjanikov, N. Gelfand, L. J. Guibas, and H. Pottmann. Dynamic geometry registration. In *Symposium on geometry processing*, pages 173–182, 2007.

[81] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[82] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[83] A. Myronenko, X. Song, and . Carreira-Perpin. Non-rigid point set registration: Coherent point drift (CPD). In *NIPS*, 2006.

[84] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[85] R. A. Newcombe, D. Fox, and S. M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE CVPR*, pages 343–352, 2015.

[86] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*, October 2011.

[87] T. Pajdla and J. Matas, editors. *The Least-Squares Error for Structure from Infinitesimal Motion*, 2004.

[88] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. 2016.

[89] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo. Tv-l1 optical flow estimation. *Image Processing On Line*, 2013:137–150, 2013.

[90] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, 2017.

[91] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha. Unsupervised deep learning for optical flow estimation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

[93] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351, pages 234–241, 2015.

[94] K. Sagi, T. Ayellet, and B. Ronen. Direct visibility of point sets. In *ACM SIGGRAPH*, 2007.

[95] G. Schindler and F. Dellaert. Probabilistic temporal inference on reconstructed 3d scenes. In *IEEE CVPR*, pages 1410–1417, 2010.

[96] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *TPAMI*, 39(4), Apr. 2017.

[97] N. Snavely, I. Simon, M. Goesele, R. Szeliski, and S. M. Seitz. Scene reconstruction and visualization from community photo collections. *Proceedings of the IEEE*, 98(8):1370–1390, 2010.

[98] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *CVPR*, 2017.

[99] H. Strasdat, R. A. Newcombe, R. F. Salas-Moreno, P. H. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. *IEEE CVPR*, 00, 2013.

[100] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, Oct. 2012.

[101] Y. Sun, M. Liu, and M. Q.-H. Meng. Improving rgb-d slam in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, 89:110 – 122, 2017.

[102] A. Taneja, L. Ballan, and M. Pollefeys. Image based detection of geometric changes in urban environments. In *IEEE ICCV*, pages 2336–2343, 2011.

[103] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. *CVPR*, 2017.

[104] Y. W. Teh. Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. 2011.

[105] S. Thrun and M. Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[106] H. Tung, H. Wei, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *NIPS*, 2017.

[107] H. F. Tung, A. W. Harley, W. Seto, and K. Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. *ICCV*, 2017.

[108] A. O. Ulusoy and J. L. Mundy. Image-based 4-d reconstruction using 3-d change detection. In *ECCV*, pages 31–45, 2014.

[109] A. Veit and S. J. Belongie. Convolutional networks with adaptive computation graphs. *CoRR*, 2017.

[110] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *CoRR*, 2017.

[111] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Kähler, D. W. Murray, S. Izadi, P. Perez, and P. H. S. Torr. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[112] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *ICRA*, pages 2043–2050, 2017.

[113] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 2004.

[114] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR*, pages 520–526, Jun 1997.

[115] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2464–2471. IEEE, 2010.

[116] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems*, 2015.

[117] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[118] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially extended kinectfusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, July 2012.

[119] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *IEEE ICCV*, 2013.

[120] F. Xue, X. Wang, S. Li, Q. Wang, J. Wang, and H. Zha. Beyond tracking: Selecting memory and refining poses for deep visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[121] J. Yan and M. Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *IEEE TPAMI*, 30(5):865–877, 2008.

[122] N. Yang, R. Wang, J. Stuckler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, pages 817–833, 2018.

[123] 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*, pages 1983–1992, 2018.

[124] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.

[125] Z. Zhang, Q. Liu, and Y. Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 2018.

[126] T. Zhou, M. Brown, N. Snavely, and D. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[127] A. Z. Zhu, W. Liu, Z. Wang, V. Kumar, and K. Daniilidis. Robustness meets deep learning: An end-to-end hybrid pipeline for unsupervised learning of egomotion. *arXiv preprint arXiv:1812.08351*, 2018.

[128] Y. Zou, Z. Luo, and J.-B. Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 36–53, 2018.