博士論文

Robust Non-rigid Tracking and Reconstruction Based on Spatial-Temporal Learning

(時空間学習に基づくロバストな非剛体追跡および再構成)

李　楊

Acknowledgements

Writing a thesis on your own is hardly feasible. Therefore, I have to thank a lot of people for their help and support.

First of all, I would like to thank my supervisor Prof. Tatsuya Harada who allowed me to join his amazing group; the Machine Intelligence Laboratory at the University of Tokyo. He continuously supported me, gave me unlimited academic freedom, and guided me through the past years. He is particularly patient, encouraging, and caring to the students and helped me overcome many difficulties. His dedication to scientific research has deeply influenced my research and working style and also shaped my attitude as a scholar. I am very grateful for having such a great advisor and mentor.

I would also like to thank Prof. Matthias Niessner who give me the chance to do an internship at his amazing lab; the Visual Computing Group at Technical University Munich. He draws my interest to the field of 4D vision. He is always available to give thorough advice and his positive way of thinking is special. He always aims for big things in research and pushed me to my full potential.

I am a strong believer in the success of joint work and collaboration. Thus, I would like to particularly thank all the fantastic co-authors of my papers: Aljaz Bozic, Tianwei Zhang, Yanli Ji, Yoshihiko Nakamura, Yoshitaka Ushiku, Tetsuo Shibuya, Matthias Niessner, and Tatsuya Harada. I am very thankful for all their contributions that helped me to successfully write this thesis. In particular, I thank Aljaz Bozic for patiently listening to and answering all my questions on non-rigid 4D reconstruction. His has a charming personality. He has always been a reliable friend and gave me valuable suggestions and encouragement when I was in difficulty.

Further, I want to thank all my colleagues at the Machine Intelligence Laboratory of the University of Tokyo for such a great working environment: Tomoyuki Takahata, Yoshitaka Ushiku, Yusuke Mukuta, Mamoru Nakamura, Yusuke Kurose, Tejero de Pablos Antonio, WESTFECHTEL Thomas, Hiroaki Yamane, Lin Gu, Hirofumi Seo, Aoi Kaneko, Miyuki Kajisa, and Sawako Maruyama. They spent tremendous effort to maintain this laboratory. I can not finish my Ph.D thesis without their support. I also thank Haowei Yeh for helping me recording the RGB-D video for the final 4D reconstruction demonstration.

A special thanks go to Bo Zheng who is heading the 3D Vision Lab at Huawei Tokyo Research Center. I did an internship at this Lab. He gave me the chance to have a look at what is going on in the industrial labs. I also get to know better about computer graphics.

Finally, I am extremely grateful to my family for their support, encouragement, and dedication to me.

Abstract

4D reconstruction of non-rigidly deforming scenes has numerous applications in computer vision, virtual/augmented reality, and robotics, etc. With the latest advancements of consumer-level depth sensors, such as Microsoft Kinect, Intel RealSense, and even smartphone mounted cameras, non-rigid reconstruction using a single RGB-D camera has gained momentum. However, due to the high complexity and non-convexity of the problem and the limitation of range sensors, a robust reconstruction system for generic non-rigidly deforming scene remains a challenge. This thesis aims to develop a single movable RGB-D camera based non-rigid reconstruction system that can 1) handle large non-rigid motion, 2) simultaneously reconstruct both the static background and the dynamic foreground objects of a given scene, 3) handle scene occlusion, and 4) yield global consistent camera pose tracking. We achieve this system by combining the classic tracking and reconstruction method with spatial-temporal priors that are learned from data. Specifically, to satisfy 1), we redefine the non-rigid tracking energy using the distance function that is defined in learned feature space. The feature is trained in the way that it can alleviate the non-convexity of this problem. To satisfy 2), we propose SplitFusion which first split the scene into rigid or non-rigid surfaces and then simultaneously performs tracking and dense reconstruction for both rigid and non-rigid components of the scene. To satisfy 3), we propose a completion approach to jointly recover the occluded structure and motion from partial RGB-D camera observation. To satisfy 4), we propose a self-supervised monocular VO that uses a pose graph optimization back-end to guarantee global camera pose consistency. Thorough experiments demonstrate the advantage of the proposed system in terms of tracking robustness and reconstruction quality in non-rigid scenes.

Table of contents

Chapter 1

Background

We live in a 4D world with three spatial dimensions and one temporal dimension. The ability to understand the 3D structure of the environment and how the structure evolves in the time axis is crucial for successful interaction with the outside world. For instance, as a nursing robot, to safely take care of an aged person in a house (i.e. predict the person's action and react accordingly), it needs to 1) know the 3D map of the house, 2) localize itself w.r.t the static part of the house, and most importantly 3) understand both the complete 3D shape of the person's body and how the whole body moves along time. This task refers to two main topics in computer vision: Simultaneous Localization and Mapping (SLAM) and Non-Rigid 4D Reconstruction. The former mainly emphasis on tracking and mapping of static/rigid scene, the later focus on reconstruction of the scene that is non-rigidly deforming.

With the arrival of Microsoft Kinect, and several other devices, e.g. such as the RealSense, Primesense, Google Tango, or even the recent smartphone mounted camera, RGB-D cameras have become available at a low price. These RGB-D cameras capture per-pixel color and depth images at adequate resolution and at real-time rates. The potential of these sensors has been quickly recognized in the visual computing community. They have boosted the development of both SLAM and non-rigid reconstruction algorithms.

KinectFusion [42, 66] is the first dense SLAM system that permits real-time, dense reconstruction of complex room-sized scenes using a single handheld Kinect depth sensor. The dense 3D reconstruction is done by tracking and incremental integration of consecutive overlapping depth maps into a volumetric representation [17] that is continuously refined. KinectFusion has demonstrated compelling real-time reconstructions results using a commodity GPU. However, like most SLAM methods, KinectFusion is designed based on the rigid environment assumption. They can not reconstruct a non-rigidly deforming scene.

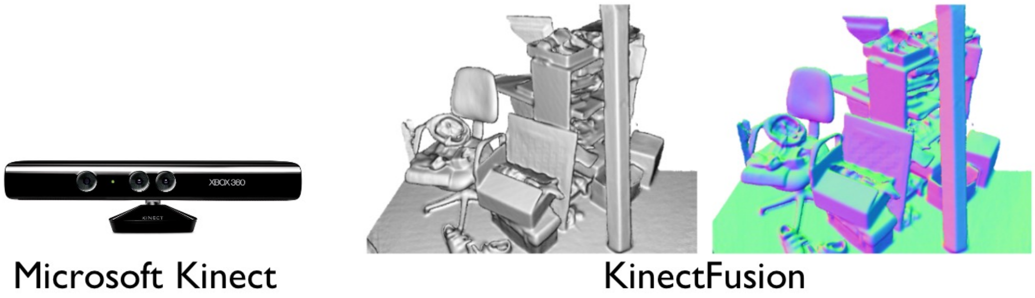


Figure 1.1 Microsoft Kinect (left), and capturing rigid scene using KinectFusion [42, 66] (right).

1.1 Non-rigid 4D reconstruction

The first approach that tackled the template-free reconstruction problem at real-time frame rates was the DynamicFusion approach by Newcombe et al. [67]. This approach enables the joint reconstruction of the geometry and motion of non-rigidly deforming objects based on a single commodity RGB-D camera. For each new input depth frame, a model-to-frame non-rigid tracking approach is used to estimate a deformation field based on a latent deformation graph [88] model. Based on the estimated deformation field, the 3D model can be non-rigidly transformed to the space of the input depth frame, which enables the update of the model based on the volumetric fusion method [17].

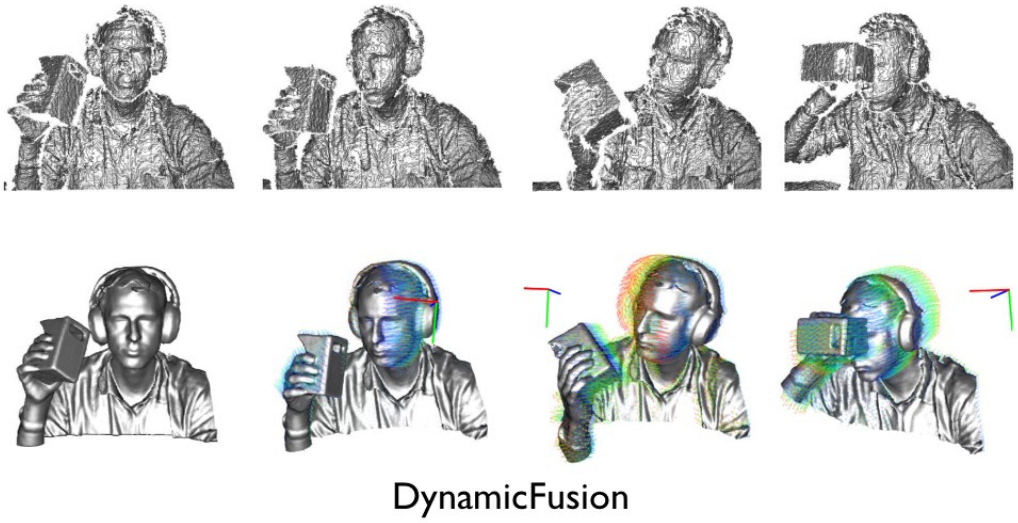


Figure 1.2 Capturing non-rigidly deforming scene using DynamicFusion [67]. Top row shows the noisy input depth map sequence. Bottom row show the reconstructed 3D model that is deformed to the corresponding depth frame.

In Figure. 1.3 we show the processing pipeline of RGB-D based non-rigid reconstruction. DynamicFusion and many other reconstruction methods basically follow this processing pipeline. Given a real-time depth map observation from a RGB-D camera, the *Non-Rigid Tracking* stage estimate the deformation that best align the 3D model and the current frame. This can be achieved in a frame-to-frame, frame-to-model, or global fashion. After the deformation is estimated, all the points from the current input frame are transformed with the estimated deformation field and are merged into the common model in the *Depth Map Fusion* stage.

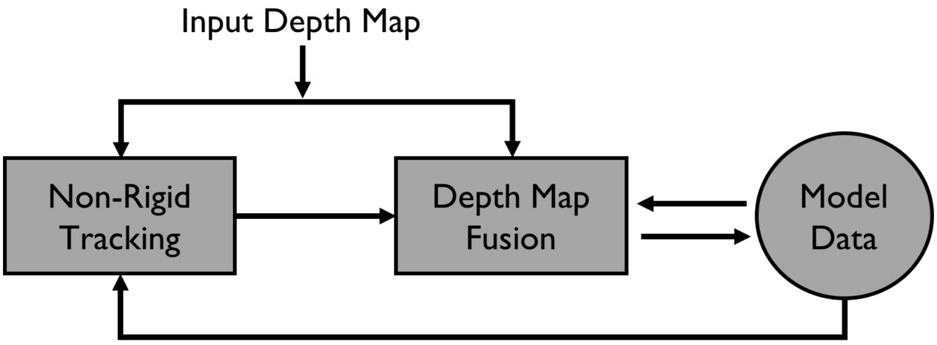


Figure 1.3 Overview of the typical RGB-D non-rigid reconstruction pipeline: first, the non-rigid tracking method aligns the input depth frame with respect to the current surface reconstruction; second, given the tracking results, the input data is integrated/fused into the current 3D model of the reconstruction.

In the rest of this section, we briefly explain the key components for dense RGB-D non-rigid reconstruction: the non-rigid tracking algorithm (cf. 1.1.1) and the depth map fusion method (cf, 1.1.2), and then discuss the fundamental challenges of this task (cf. 1.2).

1.1.1 Non-rigid tracking

At the core of the non-rigid reconstruction is the non-rigid tracking problem. The objective of non-rigid tracking is to determine the 3D motion of all points on the model given a live depth map observation. Since it is not computationally inefficient to directly model the 3D motion of all points in the scene, existing tracking algorithms usually represent the non-rigid scene with a deformable model called *Deformation Graph*. Fig. 1.4 shows an example of using deformation graph to represent a deformable scene.

Typically, non-rigid tracking is performed by a Non-Rigid Iterative Closest Point (N-ICP) algorithm [1, 72, 51, 114]. This algorithm iteratively minimizes the distance functions between correspondences points between a 3D model and a live observation. N-ICP usually employ point-to-point (euclidean distance between two points) or point-to-plane (euclidean

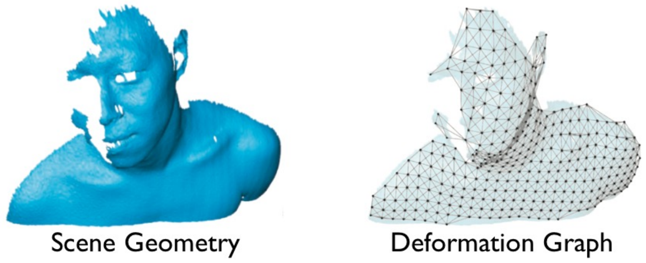


Figure 1.4 Using the deformation graph (right) to represent a deformable scene (left). The nodes of the graph are evenly sampled on the scene surface. The sampling method ensures θ-coverage of the scene, i.e., the distance of every foreground point to the nearest graph node is at most $\theta > 0$. Edges are computed between nodes based on geodesic connectivity. The figures are borrowed from [51].

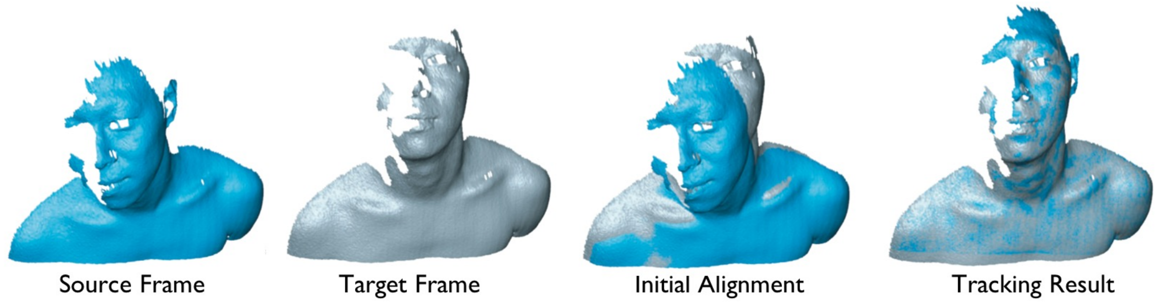


Figure 1.5 Example of performing frame-to-frame N-ICP. N-ICP iteratively deforms the source frame until the deformed source frame and the target frame are best aligned. The figures are borrowed from [51].

distance between a point and a plane) as the distance function. Besides the distance function, to prevent uncontrolled deformations and resolve motion ambiguity, the N-ICP optimization usually employ As-Rigid-As-Possible (ARAP) [85] or embedded deformation [88] as motion regularization. Such regularizers enforces the graph vertices locally move in an approximately rigid manner. Fig. 1.5 shows an example of performing N-ICP between a pair of depth frames.

1.1.2 Depth map fusion

The representation of the reconstructed model needs to be very efficient in integrating a large number of incoming depth maps. Beyond this, frame-to-model tracking requires an efficient way to generate virtual views of the model from arbitrary viewpoints to align the incoming range maps with the model, mostly using projective data association. There exist mainly two different representations to accumulate the observed RGB-D data in one common 3D model. The most commonly used way is to store the information in a regular or hierarchical 3D voxel grid, which refers to the volumetric method. Alternatively, the model can be stored as an accumulated 3D point cloud. In this thesis, we mainly consider the volumetric representation for its efficiency in extracting the scene surface.

In the volumetric representation, the voxels are associated with a distance value representing its distance to the nearest surface, where the sign of the distance indicates if the voxel is inside (-) or outside (+) of the object. The distance values that are beyond some threshold are truncated. This is the so-called Truncated Signed Distance Function (TSDF) [17]. Fig. 1.6 shows an example of using a TSDF to represent the reconstructed human model. The depth map integration is formed by the weighted average of all individual TSDFs computed for each depth map, which can be seen as de-noising the global TSDF from multiple noisy TSDF measurements. The final mesh surface of the model can be extracted using Marching Cubes.

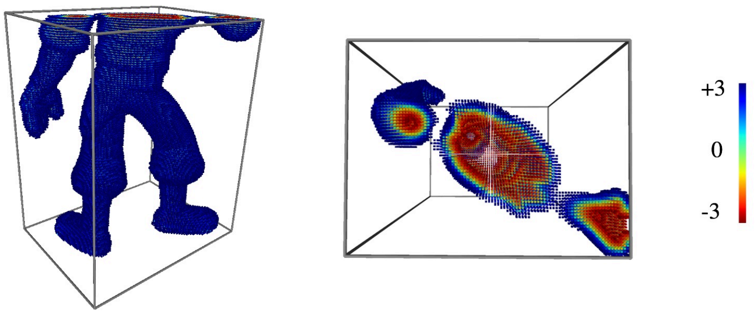


Figure 1.6 Example of volumetric truncated signed distance function with truncation of 3. Here we show a cropped TSDF of a person from side view (left) and top-down view (middle). The color of the voxeles indicate their truncation value.

1.2 Challenges of non-rigid 4D reconstruction

Although the method like DynamicFusion has shown promising results on non-rigid scene reconstructions, there are several fundamental challenges:

- **Topological Change & Scalability:** DynamicFusion parameterizes the entire scene with a single latent deformation graph as shown in Fig. 1.4. However, such a graph model inherently can not handle the topological change of the scene. Moreover, since the complexity of non-rigid tracking grows quadratically with the deformation graph's size, i.e. the number of nodes, it does not scale well to larger scenes beyond a single object.

- **Tracking failure:** At the core of non-rigid tracking is a non-linear optimization problem. However, the energy landscape of non-rigid tracking is highly non-convex.

The optimization can easily fall in undesired local minima when the target scene undergoes large non-rigid motion, which we call it as tracking failure. In theory, a perfect non-rigid tracker is sufficient for perfect non-rigid reconstruction. The other way around, bad tracking usually causes artifacts in the reconstruction.

- **Occlusion:** The fundamental limitation is that one can only obtain partial and incomplete reconstructions of a given scene because a single RGB-D camera suffers from occlusions and the physical limitations of range sensors.

Chapter 2

Toward a Versatile and Robust Non-Rigid 4D Reconstruction System

In this chapter, we clarify the scope, objective and contribution, and structure of this thesis before going into the details. In Section. 2.1, we clarify the scope of this thesis while focusing on the robustness of non-rigid reconstruction. We also show the objective reconstruction system, including 4 key properties, and our contributions toward building this system. In Section. 2.4, we show the contents of each chapters.

2.1 Scope & Objective of thesis

In this thesis, we especially tackle the task of online reconstruction of a scene that is non-rigidly deforming from a single RGB-D camera, where the camera is movable and provides real-time scene observation, and the scene is at room-scale.

Figure. 2.1 demonstrate the key directions of single RGB-D non-rigid reconstruction, while this work mainly focuses on the **Robustness** aspect of this task. More specifically, we aim to increase the reconstruction robustness under the four challenging situations: scene occlusion; large deformation; camera trajectory drift; and scene level topology change.

The goal of this thesis is to build a system that can reconstruct the room-scale environment where there are static backgrounds like walls and furniture and multiple non-rigidly deforming foreground objects such as humans and animals, using a single movable RGB-D camera. This reconstruction system should has the following features:

(i) It is robust to large non-rigid motion.

(ii) It can simultaneously reconstruct both the static background and the dynamic foreground objects.

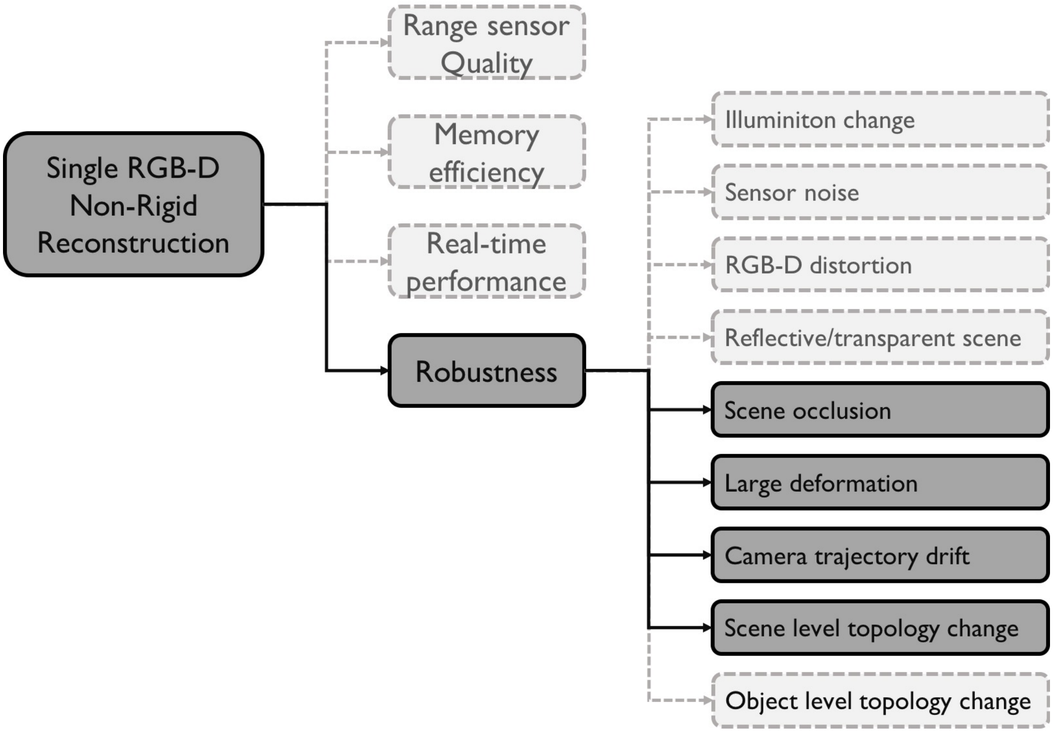


Figure 2.1 The scope of this thesis. Specifically, we aim to increase the reconstruction robustness under the four challenging situations: scene occlusion; large deformation; camera trajectory drift; and scene level topology change.

(iii) It can reconstruct the occluded part of the scene.

(iv) It can track the camera's movement w.r.t the static background, and correct the camera's trajectory drift along time.

2.2 Leveraging Spatial-Temporal Learning

In this thesis, we achieve this system by enhancing the classic tracking and reconstruction system with the deep learning-based method.

In recent years, the deep learning-based method especially the Convolutional Neural Networks (CNN) has revolutionized the field of computer vision. CNN has shown an extraordinary capacity for scene understanding from image content such as detecting and segmenting objects. It has also shown its advantage in the 3D vision and graphics field, such as tracking camera poses and predicting depth from monocular sequence [113], reconstruct 3D information from 2D image [63], and complete geometry from partial RGB-D scans [84]. Though, in theory, these tasks are all ill-posed and potentially have infinite possible solutions, the learning-based method solves these challenging tasks by learning prior knowledge from a large amount of training dataset. Since the learning-based method always finds solutions that are consistent with the data-driven knowledge, they are usually robust to outliers.

While existing methods mainly learn the representations for the static scene. We observe the fact that to leverage the learning-based method in the dynamic 4D reconstruction task, it is necessary to also learn the evolution of the data in the temporal domain. Learning from both the spatial and temporal domain, in this thesis, we refer to as spatial-temporal learning.

Spatio-temporal learning is becoming growingly important in the computer vision field with the increasing availability and importance of large spatial-temporal datasets such as RGB videos, lidar sequences, GPS trajectories of vehicles, etc. Spatio-temporal learning also has broad applications in various domains including environment and climate (e.g. wind prediction and precipitation forecasting), public safety (e.g. crime prediction), intelligent transportation (e.g. traffic flow prediction), human mobility (e.g. human trajectory pattern mining).

While the above applications focus on future prediction, in the non-rigid 4D reconstruction task, the objective of spatial-temporal learning to establish the correspondence of data given already observed RGB-D frames. More specifically, at the core of our system, we want to achieve the association of spatial scene observations across the temporal domain by tracking the motion of the scene. As shown in Fig. 2.2, in this thesis, we use neural networks to extract spatial-temporal embeding for learning non-rigid motion (c.f. Chapter. 4), semantics (c.f. Chapter. 5), shape and motion completion (c.f. Chapter. 6), and rigid motion (c.f. Chapter. 7).

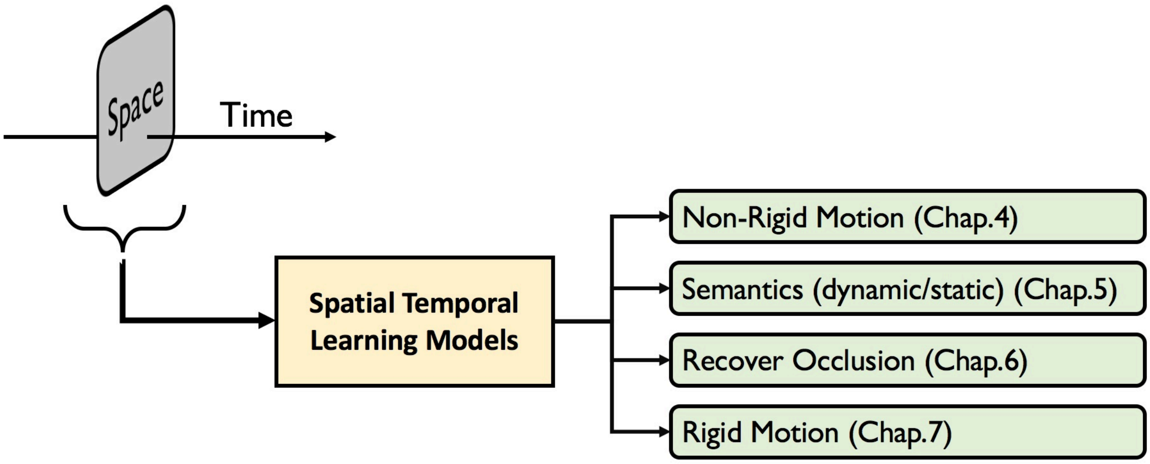

Space
Time
Spatial Temporal Learning Models
Non-Rigid Motion (Chap.4)
Semantics (dynamic/static) (Chap.5)
Recover Occlusion (Chap.6)
Rigid Motion (Chap.7)

Figure 2.2 We employ spatial-temporal learning to address the four key problems in the system.

2.3 System Overview

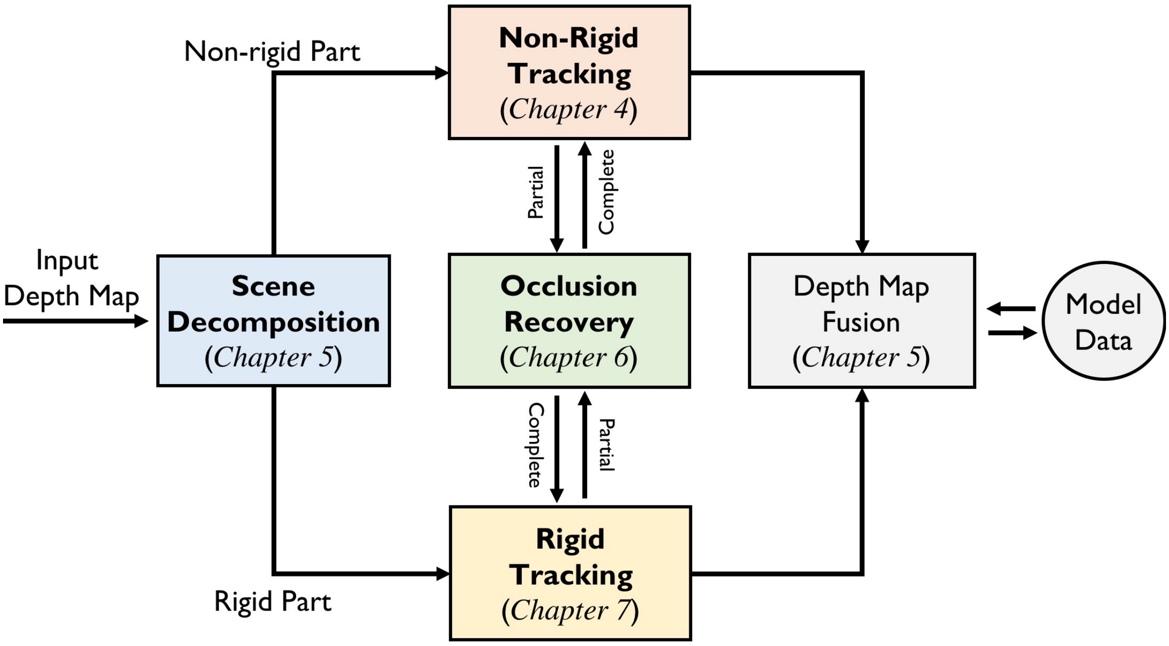

Non-rigid Part
Non-Rigid Tracking (Chapter 4)
Partial
Complete
Input Depth Map
Scene Decomposition (Chapter 5)
Occlusion Recovery (Chapter 6)
Depth Map Fusion (Chapter 5)
Model Data
Complete
Partial
Rigid Tracking (Chapter 7)
Rigid Part

Figure 2.3 The overview of the objective system. First, the input scene is decomposed into rigid or non-rigid part in the “Scene Decomposition” block (c.f. Chapter 5), the “Non-Rigid Tracking” block leverages robust learning-based tracker to align the input frame with the model data (c.f. Chapter 4), the “Occlusion Recovery” block jointly recovers the complete geometry and motion field from partial depth scans and thus also provides usefull information for both rigid and non-rigid tracking (c.f. Chapter 6), finally, the input frame is integrated into the model data (c.f. Chapter 5).

As shown in Fig. 2.2, specifically, we leverage spatial-temporal learning to achieve the system’s properties (i), (ii), (iii), and (iv) in Chapter. 4, Chapter. 5, Chapter. 6, and Chapter. 7 respectively. The full pipeline of the system in shown in Figure 2.3

Chapter. 4: Learning based Optimization for Robust Non-Rigid Tracking
we change the energy function of non-rigid tracking from the classic data fitting terms to a deep non-rigid feature fitting term. The deep non-rigid feature is learned end-to-end by convolutional neural networks through a large amount of training data. By leveraging the large receptive field of convolutional kernels and the nature of the data-driven method, the learned feature can capture the global deformation which helps to alleviate the non-convexity of the non-rigid tracking problem. The experiments show that compared to DynamicFusion, our method yield more robust tracking on large non-rigid motion.

Chapter. 5: SplitFusion
We developed a novel dense RGB-D SLAM framework called SplitFusion that simultaneously performs tracking and dense reconstruction for both rigid and non-rigid components of the scene. SplitFusion first adopts deep learning based semantic instant segmentation technique to split the scene into rigid or non-rigid surfaces. The split surfaces are independently tracked via rigid or non-rigid ICP and reconstructed through incremental depth map fusion. We found that semantic instance information informs SplitFusion to handle object-level topology change of the scene, and also make the system more scalable than DynamicFusion.

Chapter. 6: Structure and Motion Completion
we propose a data-driven completion method to jointly recover the occluded structure and their motion field from partial RGB-D frame observation. Our method not only generates good quality geometry but also yields reasonable motion fields for the occluded part of the scene.

Chapter. 7, Global consistent tracking for learned VO
we picked up the monocular visual odometry (VO) problem, for which we propose a self-supervised monocular VO that uses a pose graph optimization back-end to guarantee global camera pose consistency. We demonstrate the advantage of our method on handling camera trajectory drift. We demonstrate that spatial-temporal constraints combined yield better VO estimation. This method is useful for large scale reconstruction, this will be demonstrate in the System Demonstration chapter.

Putting all the features together in Table. 2.1, we compare our method with the state-of-the-art RGB-D based dense reconstruction method for static scene: KinectFusion [42, 66], and dynamic scene: DynamicFusion [67]. We demonstrate the full system in Chapter. 8.

	KinectFusion	DynamicFusion	Ours
Rigid tracking & Reconstruction	✓		✓
Non-Rigid tracking & Reconstruction		✓	✓
Semantic instance understanding			✓
Handle scene level topology change			✓
Scalable	✓		✓
Robust to large deformation			✓
Handle Occlusion			✓

Table 2.1 Comparison with the main-stream RGB-D camera based scene reconstruction.

We propose the following contributions toward building this system:

- A learning-based non-rigid tracking method that is robust to large non-rigid motion.
- A framework that simultaneous reconstruct both rigid and non-rigid scene elements.
- A completion method that joint recover occluded geometry and motion field from partial RGB-D observation.
- An unsupervised monocular tracking system that has a pose graph optimization back-end to ensure global pose tracking consistency.

2.4 Structure of thesis

In Chapter 1, we describe the standard pipeline of non-rigid tracking and reconstruction, and we also state the key fundamental limitations of single RGB-D camera based non-rigid scene reconstructions. In Chapter 2, we clarify the scope, objective, contribution, and structure of this thesis. In Chapter 3, we have a more comprehensive discussion on existing works. In Chapter 4, we employ learnable optimizations to improve non-rigid tracking robustness for large non-rigid motion. In Chapter 5, We present SplitFusion, a novel dense RGB-D SLAM framework that simultaneously performs tracking and dense reconstruction for both rigid and non-rigid components of the scene. In Chapter 6, we propose a completion method to handle occlusion problem. In Chapter 7, We propose an unsupervised learning based monocular visual odometry called NeuralBundler, and an Efficient loop closing procedure to correct the global camera tracking drift along time. In Chapter 8, we demonstrate the full system on a real-world sequence. Finally, in Chapter 9, we conclude this thesis and remark future directions.

Chapter 3

Related work

This chapter presents related work on static, and non-rigid tracking and reconstruction.

3.1 Rigid reconstruction using RGB-D camera

Online rigid reconstruction is directly related to Simultaneous Localization and Mapping (SLAM), which focuses on the problem of robot navigation in unknown environments; e.g., [65] [27] and many others. Dense SLAM require tracking and incremental integration of consecutive overlapping depth maps into a 3D representation that is continuously refined. Typically, camera tracking is usually performed by an Iterative Closest Point (ICP) algorithm [5]. Divided by the 3D representation types, online dense 3D reconstruction has two main groups: the *Surfel-Based* method and the *Volumetric* method. A *Surfel* [74] is a isolated 3D primitive with 3D coordinate, shape, and rendering attributes. For its simplicity and light computation, it has been applied in many reconstruction frameworks [46][99]. In *Volumetric* method, depth maps are converted into truncated signed distance function (TSDF) [17] and cumulatively averaged into a regular voxel grid. The final surface is usually extracted as the zero-value set of the implicit function using ray-casting [71]. KinectFusion [66] is the first volumetric method that demonstrated compelling real-time reconstructions using a commodity GPU. However, the use of a regular voxel grid imposes a large memory footprint, making KinectFusion impractical for large scale scene reconstruction. Nießner *et al.*[69] addressed the memory issue by introducing the hierarchical spatial voxel hashing method. Dai *et al.* [19] extends volumetric method into a complete SLAM system called BundleFusion, with a loop-closure, global pose optimization, and model update back-end.

3.2 Non-rigid tracking and reconstruction with RGB-D camera

The SLAM methods assume that scenes are static. However, dynamic elements in the scene can cause trouble for camera 6-dof pose tracking, leading to artifacts in the reconstruction. The straightforward solution is to remove the dynamic objects based on object detection and then apply the static SLAM solutions. For instance, PoseFusion uses skeleton tracking and geometry clusters to segment out humans; DS-SLAM [103] applied SegNet to detect and remove foreground humans and then estimate the camera motion with ORB-SLAM2 [64].R. Martin *et al.*proposed Co-Fusion [77] and MaskFusion [78] which apply object-level labels to track and reconstruct rigid objects, but they cannot handle non-rigid objects. Some researchers insisted to find out the dynamic clusters from the dense RGB-D point clouds. StaticFusion [80] jointly refine the Visual Odometry (VO) and dynamic segmentation using intensity and depth residuals. Zhang *et al.*[110] proposed to involve optical flow residuals to segment non-rigid clusters from the dense point cloud clusters. Although these approaches show improvements in ego-motion estimation and background reconstruction, in many practical applications, the dynamic elements are the most important targets in the scene. In particular, humans are widely encountered and can change the state of the scene in many ways. As such, in scenarios such as autonomous driving or the deployment of robots in areas with many humans, it would not be appropriate to discard moving elements.

The general solution for reconstruction in dynamic environments is to explicitly model the scene non-rigidity. The reconstruction of general non-rigidly deforming objects/scenes based on real-time depth sensor data has a long tradition [94]. One of the traditional methods use the pre-scanned template and transforming the template to live frames from RGB-D or stereo camera data. The first approach that tackled the template-free reconstruction problem at real-time frame rates was the DynamicFusion approach by Newcombe et al. [67]. This approach enables the joint reconstruction of object geometry and motion based on a single commodity depth camera, e.g. the Microsoft Kinect. Zollhöfer *et al.* [41] proposed VolumeDeform, which employs a flip-flop data-parallel optimization schema that tackles the non-rigid registration at real-time rates. Besides, they improve tracking robustness using global sparse correspondence from hand-engineered feature matching. SurfelWarp [30] reduced the memory and computation cost by employs Surfel rather than a volume as the 3D representation. Using multiple RGB-D cameras, Fusion4D [25] achieves high quality reconstruction results at the cost of the more complicated hardware setup.

The core of non-rigid reconstruction method is the non-rigid tracking algorithm. Typically, non-rigid tracking is usually performed by a Non-Rigid Iterative Closest Point (N-ICP) algorithm [1, 72, 51, 114], where the point-to-point or point-to-plane distance of correspondences points are iteratively minimized. To prevent uncontrolled deformations and resolve motion ambiguity, the N-ICP optimization usually employ As-Rigid-As-Possible (ARAP) [85] or embedded deformation [88] as motion regularization. The seminal DynamicFusion and VolumeDeform tipically employ the classic N-ICP algorithms for tracking. DoubleFusion [104] demonstrated the advantage of using a shape prior (*i.e.*the SMPL [58] model) in human tracking. Physical constraints have also been explored in [105, 98]. KillingFusion [83] directly estimate the motion field given a pair of Signed Distance Field (SDF).

3.3 Deep learning for non-rigid tracking and reconstruction

This line of research focuses on solving motion tracking and reconstruction tasks from a deep learning perspective. One of the promising ideas is to replace the hand-engineered descriptors with the learned ones. For instance, the Learned Invariant Feature Transform (LIFT) is proposed in [102], the volumetric descriptor for 3D matching is proposed in [106], and the coplanarity descriptor for plane matching is proposed in [81]. For non-rigid localization/tracking, Schmidt *et al.* [79] use Fully-Convolutional networks to learn the dense descriptors for upper torso and head of the same person; The recent works DeepDeform [9] replaces the noisy hand-engineered feature correspondence by more robust neural network based correspondence matching that is learned from a large amount of labeled non-rigid sequence. Regression networks have also been used to directly map input sensor data to motions, including the camera pose tracking [112], the dense optical flow tracking [24], and the 3D scene flow estimation [55].

The problem of motion regression is that the regressors could be overwhelmed by the complexity of the task, therefore, leading to severe over-fitting. A more elegant way is to let the model focus on a simple task, such as feature extraction while using classic optimization tools to solve the rest. This resulted in the recent works that combine Gauss-Newton optimization and deep learning to learn the most suitable features for image alignment [14], pose registration [39, 61], and multi-frame direct bundle-adjustment [90]. Optical/scene flow [24, 92, 89, 96, 55, 100, 56, 37, 60] is a closely related technique. They have been used to generate initial guess for non-rigid tracking in [25, 30, 8, 26, 97]. Among these works, FlowNet3D [55] is the first method that directly estimates scene flow from two sets of point clouds.

Chapter 4

Learning based Optimization for Robust Non-Rigid Tracking

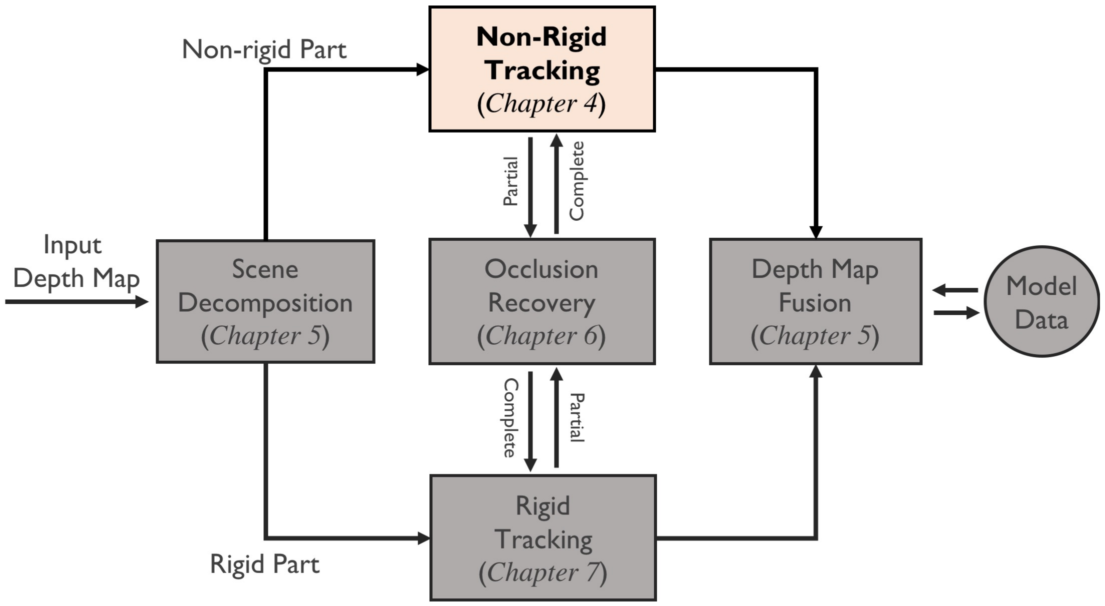

Non-rigid Part
Non-Rigid Tracking (Chapter 4)
Partial
Complete
Input Depth Map
Scene Decomposition (Chapter 5)
Occlusion Recovery (Chapter 6)
Depth Map Fusion (Chapter 5)
Model Data
Complete
Partial
Rigid Tracking (Chapter 7)
Rigid Part

Figure 4.1 The focus of this chapter and its position in the entire system.

As shown in Figure. 4.1, this chapter focuses on the core problem of the 4D reconstruction system: non-Rigid tracking. Here we aim to develop a non-rigid tracking algorithm that is robust to large deformation. One of the widespread solutions for non-rigid tracking has a nested-loop structure: with Gauss-Newton to minimize a tracking objective in the outer loop, and Preconditioned Conjugate Gradient (PCG) to solve a sparse linear system in the inner loop. In this chapter, we employ learnable optimizations to improve tracking robustness and speed up solver convergence. First, we upgrade the tracking objective by integrating an alignment data term on deep features which are learned end-to-end through CNN. The new tracking objective can capture the global deformation which helps Gauss-Newton to jump over local minimum, leading to robust tracking on large non-rigid motions. Second, we bridge the gap between the preconditioning technique and learning method by introducing a ConditionNet which is trained to generate a preconditioner such that PCG can converge within a small number of steps. Experimental results indicate that the proposed learning method converges faster than the original PCG by a large margin.

4.1 Introduction

Non-rigid dynamic objects, *e.g.*humans and animals, are important targets in both computer vision and robotics applications. Their complex geometric shapes and non-rigid surface changes result in challenging problems for tracking and reconstruction. In recent years, using commodity RGB-D cameras, the seminal works such as DynamicFusion [67] and VolumeDeform [41] made their efforts to tackle this problem and obtained impressive non-rigid reconstruction results. At the core of DynamicFucion and VolumeDeform are non-linear optimization problems. However, this optimization can be slow, and can also result in undesired local minima. In this chapter, we propose a learning-based method that finds optimization steps that expand the convergence radius (*i.e.*avoids local minima) and also makes convergence faster. We test our method on the essential inter-frame non-rigid tracking task, *i.e.*to find the deformation between two RGB-D frames, which is a high-dimensional and non-convex problem. The absence of an object template model, large non-overlapping area, and observation noise in both source and target frame make this problem even more challenging. This section will first review the classic approach and then put our contributions into context.

Non-rigid Registration The non-rigid surface motions can be roughly approximated through the “deformation graph" [88]. In this deformable model, all of the unknowns, *i.e.*the rotations and translations, are denoted as $\mathcal{G}$. Given two RGB-D frames, the goal of non-rigid registration is to determine the $\mathcal{G}$ that minimizes the typical objective function:

$$\min_{\mathcal{G}}\{\mathbf{E}_{fit}(\mathcal{G})+\lambda\mathbf{E}_{reg}(\mathcal{G})\} \tag{4.1}$$

where $\mathbf{E}_{fit}$ is the data fitting term that measures the closeness between the warped source frame and the target frame. Many different data fitting terms have been proposed over the past decades, such as the geometric point-to-point and point-to-plane constraints [51, 114, 67, 41] sparse SIFT descriptor correspondences [41], and the dense color term [114], *etc*. The term $\mathbf{E}_{reg}$ regularizes the problem by favoring locally rigid deformation. Coefficient λ balances these two terms. The energy (4.1) is minimized by iterating the Gauss-Newton update step [6] till convergence. Inside each Gauss-Newton update step a large linear system needs to be solved, for which an iterative preconditioned conjugate gradient (PCG) solver is commonly used.

This classic approach cannot properly handle large non-rigid motions since the data fitting term $\mathbf{E}_{fit}$ in the energy function (4.1) is made of local constraints (*e.g.*dense geometry or color maps), which only work when they are close to the global solution, or global constraints that are prone to noise (*e.g.*sparse descriptor). In the case of large non-rigid motions, these constraints cannot provide convergent residuals and lead to tracking failure. See Fig. 4.2 for the failure and sucess case of non-rigid tracking.

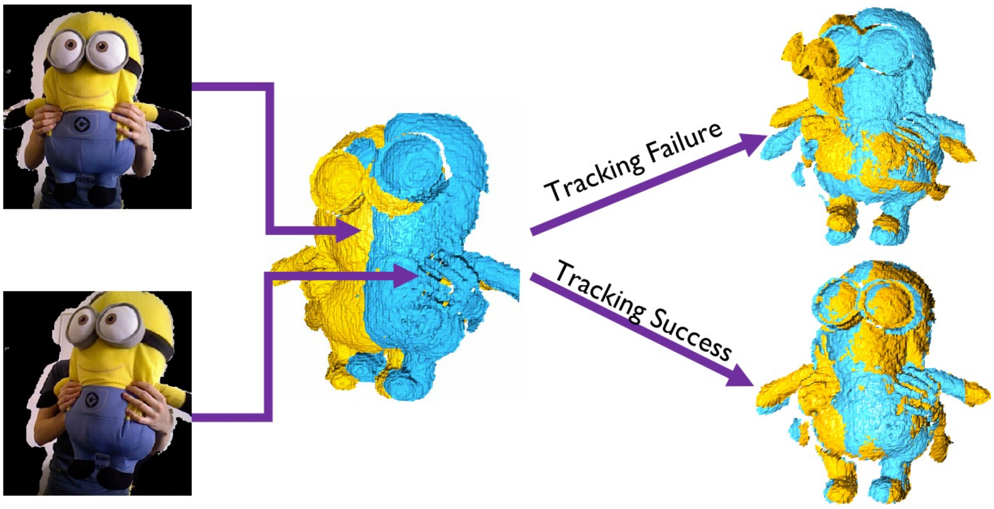


Figure 4.2 A failure and success case of non-rigid tracking, the image examples are obtained from VolumeDeform [41]. For the high non-convexity of this problem, the tracking is easily converged to bad local minima and results tracking failure.

In this chapter, we alleviate the non-convexity of this problem by introducing a deep feature alignment term into $\mathbf{E}_{fit}$. The deep features are extracted through an end-to-end trained CNN. We assume that, by leveraging the large receptive field of convolutional kernels and the nature of the data-driven method, the learned feature can capture the global information which helps Gauss-Newton to jump over local minimums. Fig. 4.3 shows the information involves in computing the image gradient and the CNN feature map gradient, which guides the optimization of non-rigid tracking.

As illustrated in Fig. 4.4, preconditioning speeds up the convergence of an iterative solver. The general idea behind preconditioning is to use a matrix, called *preconditioner*,

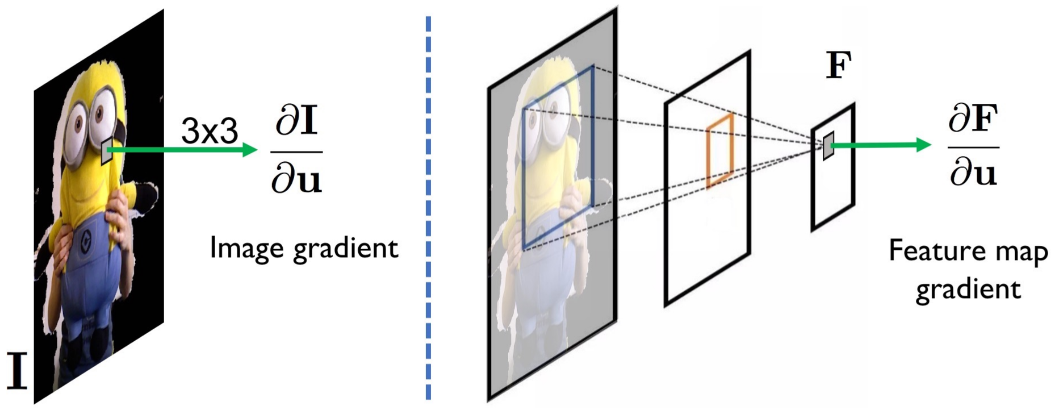

I
3x3
$\frac{\partial \mathbf{I}}{\partial \mathbf{u}}$
Image gradient
F
$\frac{\partial \mathbf{F}}{\partial \mathbf{u}}$
Feature map gradient

Figure 4.3 The information involves in computing the image gradient (left) and the CNN feature map gradient (right). The gradient of a pixel only captures the information from a small 3×3 local patch thus not suitable for global non-rigid alignment. The CNN feature has a large receptive field, therefore, capture the global information of the scene.

to modify an ill-posed system into a well-posed one that is easier to solve. As the hard-coded block-diagonal preconditioner was not designed specifically for the non-rigid tracking task, the existing non-rigid tracking solvers are still time-consuming. We argue that PCG converges much faster if the design of the preconditioner involves prior expert knowledge of this specific task. Then we raise the question: *does the data-driven method learn a good preconditioner?* In this chapter, we exploit this idea by training neural network to generate a preconditioner such that PCG can converge within a few steps.

Our contribution is twofold:

- We introduce a deep feature fitting term based on end-to-end learned CNN for the non-rigid tracking problem. Using the proposed data fitting term, the non-rigid tracking Gauss-Newton solver can converge to the global solution even with large non-rigid motions.
- We propose ConditionNet that learns to generate a problem-specific preconditioner using a large number of training samples from the Gauss-Newton update equation. The learned preconditioner increases PCG's convergence speed by a large margin.

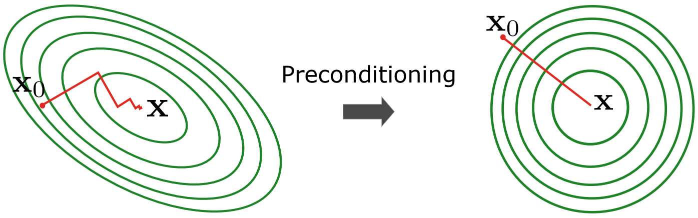


Figure 4.4 Example of using the Deepest Decent to solve a 2D system. Deepest Decent needs multiple steps to converge on an ill-conditioned system (left) and only one step on a perfectly conditioned system (right). Intuitively, Preconditioning is trying to modify the energy landscape from an elliptical paraboloid into a spherical one such that from any initial position, the direction of the first-order derivative directly points to the solution.

4.2 Method

4.2.1 Learning deep non-rigid feature

Scene representation

The input of our method is two frames that are captured using a commodity RGB-D sensor. Each frame contains a color map and a depth map both at the size of 640×480. Calibration was done to ensure that color and depth were aligned in temporal and spatial domain. We denote the source frame as **S**, and the target frame as **T**.

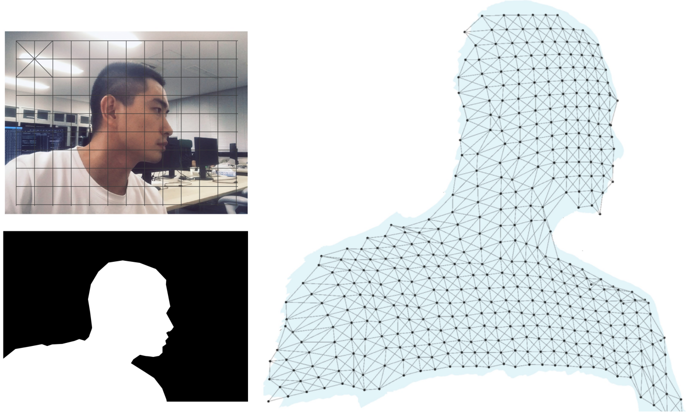


Figure 4.5 Our deformation graph. Left top: Uniform sampling on the pixel grid. Let bottom: Binary mask acquired using simple depth threshold or depth aided human annotation. Right: Masked 3D deformation graph.

We approximate the surface deformation with the deformation graph $\mathscr{G}$. Fig. 4.5 shows an example of our deformation graph. We uniformly sample the image, resulting in a rectangle mesh grid of size $w \times h$. A point in the mesh grid is treated as a node in the deformation graph. Each node connects exactly to its 8 neighboring nodes. To filter out the invalid nodes, a binary mask $\mathscr{V} \in \mathbb{R}^{w \times h}$ is constructed by checking if the node is from the background, holds invalid depth, or lies on occlusion boundaries with large depth discontinuity. Similarly, edges are filtered by the mask $\mathscr{E} \in \mathbb{R}^{w \times h \times 8}$ if they link to invalid nodes or go beyond the edge length threshold. In the deformation graph, the node i is parameterized by a translation vector $\mathbf{t}_i \in \mathbb{R}^3$ and a matrix $\mathbf{R}_i \in \mathrm{SO3}$. Putting all parameters into a single vector, we get

$$\mathscr{G} = \{\mathbf{R}_i, \mathbf{t}_i|_{i=1,2,\cdots,w \times h}\}$$

Deep Feature Fitting Term

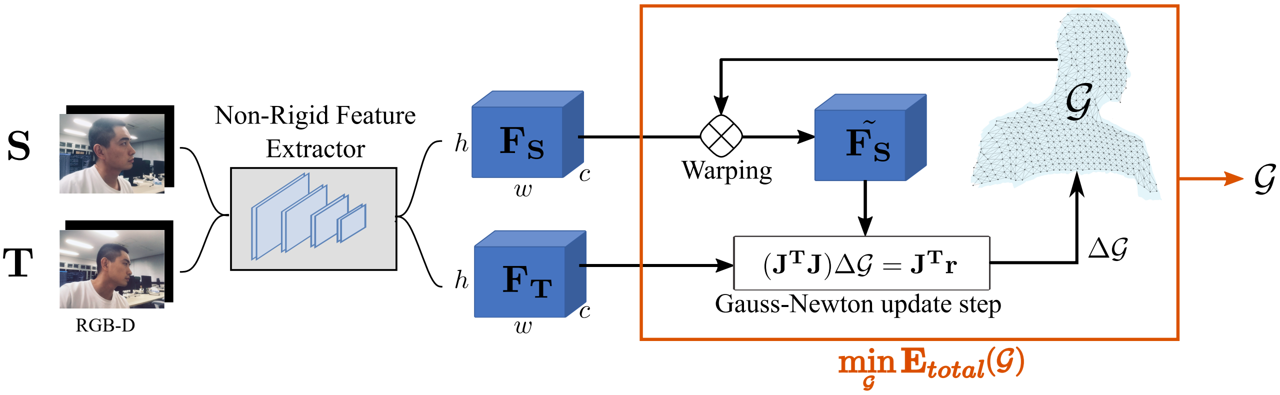


Figure 4.6 High-level overview of our non-rigid feature extractor training method. Jacobian $\mathbf{J}$'s entries for the feature term (4.4) can be precomputed according to the inverse composition algorithm. Other entries in Jacobian $\mathbf{J}$ and all entries in residues $\mathbf{r}$ are recomputed in each Gauss-Newton iteration. For simplicity, the geometric fitting term (5.5) and regularization term (5.6) are omitted from this figure.

We use the function $\mathscr{F}(\cdot)$, which is based on fully convolutional networks [57], to extract feature map from source frame $\mathbf{S}$ and target frame $\mathbf{T}$. The encoded feature maps are:

$$\mathbf{F_S} = \mathscr{F}(\mathbf{S}), \quad \mathbf{F_T} = \mathscr{F}(\mathbf{T}) \tag{4.2}$$

We apply up-sampling layers in the neural network such that the encoded feature map has the size $w \times h \times c$, where c is the dimension of a single feature vector. Thus the feature map and the deformation graph have the same rows and columns. This means that a feature vector and a graph node have a one-to-one correspondence (to reduce GPU memory overhead and speed up the learning). We denote $\mathbf{D_S} \in \mathbb{R}^{w \times h}$ and $\mathbf{D_T} \in \mathbb{R}^{w \times h}$ as the sampled depth map from source and target frames. Given the translation vectors $\mathbf{t}_i \in \mathscr{G}$, and the depth value

$\mathbf{D_T}(i)$, the projected feature for the pixel i can be obtained by

$$\tilde{\mathbf{F}}_\mathbf{S}(i) = \mathbf{F_S}(\pi(\mathbf{t}_i, \mathbf{D_T}(i))) \tag{4.3}$$

where $\pi(\cdot) : \mathbb{R}^2 \to \mathbb{R}^2$ is the warping function that maps one pixel coordinate to another pixel coordinate by applying translation $\mathbf{t}_i$ to a back-projected pixel i, and projecting the transformed point to the source camera frame. The warped coordinate are continuous values. $\tilde{\mathbf{F}}_\mathbf{S}(i)$ is sampled by bi-linearly interpolating the 4 nearest features on the 2D mesh grid. This sampling operation is made differentiable using the spatial transformer network defined in [43]. Then the **deep feature fitting term** is defined as

$$\mathbf{E}_{fea}(\mathscr{G}) = \lambda_f \sum_{i=0}^{w\times h} \mathscr{V}_i \cdot ||\tilde{\mathbf{F}}_\mathbf{S}(i) - \mathbf{F_T}(i)||^2 \tag{4.4}$$

Note that compared to the classic color-consistency constraints, the learned deep feature captures high-order spatial deformations in the scans, by leveraging the large receptive field size of the convolution kernels.

Total Energy

To resolve the ambiguity in Z axis, we adopt a projective depth, which is a rough approximation of the point-to-plane constraint, as our **geometric fitting term**. This term measures the difference between warped depth map $\tilde{\mathbf{D}}_\mathbf{S}$ and the depth map of target frame. It is defined as

$$\mathbf{E}_{geo}(\mathscr{G}) = \lambda_g \sum_{i=0}^{w\times h} \mathscr{V}_i \cdot ||\tilde{\mathbf{D}}_\mathbf{S}(i) - \mathbf{D_T}(i)||^2 \tag{4.5}$$

Finally, we regularize the shape deformation by the ARAP **regularization term**, which encourages locally rigid motions. It is defined as

$$\mathbf{E}_{reg}(\mathscr{G}) = \lambda_r \sum_{i=0}^{w\times h} \sum_{j\in\mathscr{N}_i} \mathscr{E}_{i,j} \cdot ||(\mathbf{t}_i - \mathbf{t}_j) - \mathbf{R}_i(\mathbf{t}_i^{'} - \mathbf{t}_j^{'})||^2 \tag{4.6}$$

Where $\mathscr{N}_i$ denotes node-i's neighboring nodes, and $\mathbf{t}_j^{'}$, $\mathbf{t}_j^{'}$ are the positions of i, j after the transformation. To summarize the above, we obtain the following energy for non-rigid tracking:

$$\mathbf{E}_{total}(\mathscr{G}) = \mathbf{E}_{fea}(\mathscr{G}) + \mathbf{E}_{geo}(\mathscr{G}) + \mathbf{E}_{reg}(\mathscr{G}) \tag{4.7}$$

The three terms are balanced by $[\lambda_f, \lambda_g, \lambda_r]$. The total energy is then optimized by the Gauss-Newton update steps:

$$(\mathbf{J}^{\mathbf{T}}\mathbf{J})\Delta\mathscr{G} = \mathbf{J}^{\mathbf{T}}\mathbf{r} \tag{4.8}$$

where **r** is the error residue, and **J** is the Jacobian of the residue with respect to $\mathscr{G}$. This equation is further solved by the iterative PCG solver.

4.2.2 Differentiable non-rigid optimization

The learning pipeline is shown in Fig. 4.6. We integrate all energy optimization steps into an end-to-end training pipeline. To this end, we need to make both Gauss-Newton and PCG differentiable. In the Gauss-Newton case, the update steps stop when a specified threshold is reached. Such if-else based termination criteria prevents error back-propagation. We apply the same solution as in [61, 90, 39], *i.e.*we fix the number of Gauss-Newton iterations. In this project, we set this number to a small digit. There are two reasons behind this: 1) For the recursive nature of the Gauss-Newton layer, large iterations number will induce instability to the network training, 2) By limiting the available step, the feature extractor is pushed to produce the features that allow Gauss-Newton solver to make bigger jumps toward the solution. Thus we can achieve faster convergence and robust solving.

Back-propagation through PCG can is done in a different fashion as described in [3]. Equation (5.7) need to be solved in every Gauss-Newton iteration. Let's represent $\mathbf{J}^{\mathbf{T}}\mathbf{J}$ by **A**, $\Delta\mathscr{G}$ by **x**, and $\mathbf{J}^{\mathbf{T}}\mathbf{r}$ by **b**, then we get the following iconic equation:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.9}$$

Suppose that we have already got the gradient of loss $\mathbb{L}$ *w.r.t* to the solution **x** as $\partial\mathbb{L}/\partial\mathbf{x}$. We want to back propagate that quantity onto **A** and **b**:

$$\frac{\partial\mathbb{L}}{\partial\mathbf{b}} = \mathbf{A}^{-1}\frac{\partial\mathbb{L}}{\partial\mathbf{x}} \tag{4.10}$$

$$\frac{\partial\mathbb{L}}{\partial\mathbf{A}} = (-\mathbf{A}^{-1}\frac{\partial\mathbb{L}}{\partial\mathbf{x}})(\mathbf{A}^{-1}\mathbf{b})^{\mathrm{T}} = -\frac{\partial\mathbb{L}}{\partial\mathbf{b}}\mathbf{x}^{\mathrm{T}} \tag{4.11}$$

which means that back-propagating through the linear system only need another PCG solve for Equation (4.10).

4.2.3 Training objective & data acquisition

The method outputs the final deformation graph after a few Gauss-Newton iterations. We apply the L1 flow loss on all the translation vectors $\mathbf{t}_i \in \mathscr{G}$ in the deformation graph

$$\mathbb{L}_{flow} = \sum_{\mathbf{t}_i \in \mathscr{G}} |\mathbf{t}_i - \mathbf{t}_{i,gt}| \tag{4.12}$$

where $\mathbf{t}_{i,gt} \in \mathbb{R}^3$ is node-i's ground truth 3D translation vector, *i.e.*the scene flow.

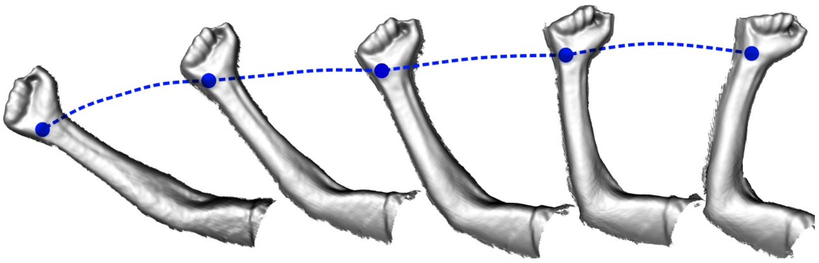


Figure 4.7 Using our non-rigid tracking and reconstruction method to obtain point-point correspondence. This method can generate accurate correspondence when the motion is small. The long term correspondence between distant frames can be obtained by accumulating small inter-frame motions through time and space.

Collecting $\mathbf{t}_{i,gt}$ is a non-trivial task. Inspired by Zeng *et al.*[106] and Schmidt *et al.* [79], we realize that the 3D correspondence ground truth can be achieved by running the state-of-the-art tracking and reconstruction methods such as BundleFusion [19], for rigid scenes, or DynamicFusion [67]/ VolumeDeform [41], for non-rigidly deforming scenes. For the rigid training set, we turn to the ScanNet, which contains a large number of indoor sequences with BundleFusion based camera trajectory. For the non-rigid training dataset, as shown in Fig. 4.7, we run our geometry based non-rigid reconstruction method (which is similar to DynamicFusion [67]) on the collected non-rigid sequences. We argue that non-rigid feature learning could benefit from rigid scenes. Since the rigid scenes can be considered as a subset of the non-rigid ones, the domain gap is not that huge when we approximate the rigid object surface from a deformable perspective. Eventually, the feature learning pipeline is pre-trained on ScanNet and fine-tuned on our non-rigid dataset.

4.2.4 Data-driven preconditioner

Preconditioning as a method of transforming a difficult problem into one that is easier to solve has centuries of history. Back to 1845, the Jacobi's Method [13] was first proposed to improve the convergence of iterative methods. Block-Jacobi is the simplest form of preconditioning, in which the preconditioner is chosen to be the block diagonal of the linear system that we want to solve. Despite its easy accessibility, we found that applying it shows only a marginal improvement in our problem. Other methods, such as Incomplete Cholesky Factorization, multiGrid method [91] or successive over-relaxation [93] method have shown their effectiveness in many applications. In this section, we exploit the potential of data-driven preconditioner to solve the linear system in the non-rigid tracking task.

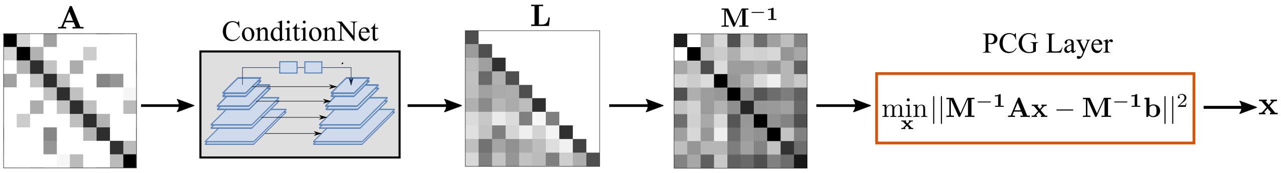

A
ConditionNet
L
M⁻¹
PCG Layer
$\min_{\mathbf{x}} ||\mathbf{M^{-1}Ax} - \mathbf{M^{-1}b}||^2$
x

Figure 4.8 Overview of ConditionNet-Dense. The output **L** is the lower triangle matrix of the preconditioner. After a few iterations in the PCG layer, the solution **x** is then penalized by the L1 loss (4.16). The whole pipeline can be trained end-to-end.

Preconditioner $\mathbf{M^{-1}}$ modifies the system $\mathbf{Ax} = \mathbf{b}$ to

$$\mathbf{M^{-1}Ax} = \mathbf{M^{-1}b} \tag{4.13}$$

which is easier to solve. From the iterative optimization perspective, solving (4.13) is equal to finding the $\mathbf{x}$ that minimizes the quadratic form

$$\min_{\mathbf{x}} ||\mathbf{M^{-1}Ax} - \mathbf{M^{-1}b}||^2 \tag{4.14}$$

Here, we propose the ConditionNet $\mathscr{C}(\cdot)$ based on neural networks with an encoder and decoder structure to do the mapping:

$$\mathscr{C}(\cdot) : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n} : \mathbf{A} \to \mathbf{M^{-1}}$$

A good preconditioner should be a symmetric positive definite (SPD) matrix, otherwise, the PCG can not guarantee to converge. To this end, the ConditionNet first generates the lower triangle matrix **L**. Then the preconditioner $\mathbf{M^{-1}}$ is computed as

$$\mathbf{M^{-1}} = \mathbf{LL^T} \tag{4.15}$$

Empirically, we apply a hard positive threshold on $\mathbf{M^{-1}}$'s diagonal entries to combat the situations that there exist zero singular values. By doing this, $\mathbf{M^{-1}}$ is ensured to be an SPD matrix in our case.

The matrix density, *i.e.*the ratio of non-zero entries, play an important role in preconditioning. On one end, a denser preconditioner has a higher potential to approximate $\mathbf{A^{-1}}$, which is the perfect conditioner, but the matrix inverse itself is time-consuming. On the other end, a sparser matrix is cheaper to achieve while leading to a poor preconditioning effect. To examine the trade-off between efficiency and effectiveness, we propose the following three ConditionNet variants. They use the same network structure but generate preconditioners with different density, from dense to sparse.

ConditionNet-Dense. As shown in Fig. 4.8, this one uses full matrix $\mathbf{A}$ as input and generate the dense preconditioner, in which all entries can be non-zero. Intuitively, this model is trying to approximate the perfect conditioner $\mathbf{A^{-1}}$.

ConditionNet-Sparse. This one inputs full matrix $\mathbf{A}$. For the output, a binary mask is applied such that any entry in $\mathbf{L}$ is set to zero if the corresponding entry in $\mathbf{A}$ is also zero.

ConditionNet-Diagonal. The input and output are the block diagonals of the matrices. There are $w \times h$ diagonal blocks and each block is 6×6. Since each block is directly related to a feature in the 2D mesh grid, we reshape the input block diagonal entries to a $[w,h,36]$ volume to leverage such 2D spatial correlations. The output volume is $[w,h,21]$ for the lower triangle matrix $\mathbf{L}$. This model generates the sparsest preconditioner.

Self-Supervised Training

The straight forward way to train the ConditionNet is to minimize the condition number $\kappa(\mathbf{M^{-1}A}) = \lambda_{\max}/\lambda_{\min}$ *i.e.*the ratio of the maximum and minimum singular value in $\mathbf{M^{-1}A}$. However, the time consuming singular value decomposition (SVD) makes large scale training impractical.

Instead, we propose the PCG-Loss for training. As shown in Fig. 4.8 the learned preconditioner $\mathbf{M^{-1}}$ is fed to a PCG layer to minimize (4.14) and output the solution $\mathbf{x}$. Training data generation for the ConditionNet is fully automatic; i.e., no annotations are needed to find the ground truth solution $\mathbf{x}_{gt}$ to equation $\mathbf{Ax} = \mathbf{b}$, which we do by running a standard PCG solver. To obtain $\mathbf{x}_{gt}$, the standard PCG is executed as many iterations as possible till convergence. Then the L1 PCG-Loss is applied on the predicted solution

$$\mathbb{L}_{pcg} = |\mathbf{x} - \mathbf{x}_{gt}| \tag{4.16}$$

The training samples, *i.e.*the [$\mathbf{A}$, $\mathbf{b}$] pairs, are collected from the Gauss-Newton update step in Eqn. (5.7).

During the training phase, we limit the number of available iterations in the PCG layer. This is to encourage the ConditionNet to generate a better preconditioner that achieves the same solution while using fewer steps. At the early phase of the training, the PCG layer with limited iterations does not guarantee a good convergence. The back-propagation strategy described in Section 4.2.2 can not be applied here, because incomplete solving results in wrong gradient. Instead, we directly flow the gradient through all PCG iterations for ConditionNet training.

We train ConditionNet and the non-rigid feature extractor separately. They are used together at the testing phase.

4.3 Experimental results

Implementation Details: The resolution of the deformation graph is 16 × 12. Empirically, the weighting factor $[\lambda_f, \lambda_g, \lambda_r]$ in the energy function (5.4) are set to $[1, 0.5, 40]$. The number of Gauss-Newton iterations is 3 for non-rigid feature extractor training. The number of PCG iterations is 10 for ConditionNet Training. We implement our networks using the publicly available Pytorch framework and train it with Tesla P100 GPUs. We trained all the models from scratch for 30 epochs,with a mini-batch size of 4 using Adam [48] optimizer, where $\beta_1 = 0.9$, $\beta_1 = 0.999$. We used an initial learning rate of 0.0001 and halve it every 1/5 of the total iterations.

Datasets

ScanNet ScanNet [18] is a large-scale RGBD video dataset containing 1,513 sequences in 706 different scenes. The sequences are captured by iPad Mounted RGBD sensors that provide calibrated depth-color pairs of VGA resolution. The 3D camera poses are based on BundleFusion [19]. The 3D dense motion ground truth on the ScanNet is obtained by projecting point cloud via depth and 6-Dof camera pose. We apply the following filtering process for training data. To narrow the domain gap with the non-rigid dataset, we filter out images if more than 50% of the pixels have the invalid depth or depth values larger than 2 meters. To avoid image pairs with large pose error, we filter image pairs with a large photo-consistency error. Finally, we remove the image pairs with less than 50% “covisibility", *i.e.*the percentage of the pixels that are visible from both images. Similarly, the sequences are subsampled using the intervals [2, 4, 8, 16]. We use 60k frame pairs in total and split train/valid/test as 8/1/1.

Non-Rigid Dataset We use the non-rigid dataset from Aljaž *et al.* [9] which consists of 400 non-rigidly deforming scenes, over 390,000 RGB-D frames. The distance of the objects to the camera center lies in the range [0.5m, 2.0m]. Depending on the complexity of the scene, the foreground object masks are either obtained by a simple depth threshold or depth map aided human annotation. We run our tracking and reconstruction method to obtain the ground truth non-rigid motions. We remove the drifted sequences by manually checking the tracking quality of the reconstructed model. The example of this dataset can be found in the paper [9]. Similarly to the rigid case, we sub-sample the sequences using the frame jumps [2, 4, 8, 16] to simulate the different magnitude of non-rigid deformation. For data-augmentation, we perform horizontal flips, random gamma, brightness, and color shifts for input frame pairs. Finally, we got 8.5k frame pairs in total and split train/valid/test as 8/1/1.

4.3.1 Non-rigid tracking evaluation

Baselines We implement a few variants of the non-rigid ICP (N-ICP) methods. They apply different energy terms as shown in Tab. 1. Among them, N-ICP-1 is our implementation of the method DynamicFusion [67], and N-ICP-2 is our implementation for the method described in [114]. The original two papers are focusing on the model to frame tracking problems where the model is either reconstructed on-the-fly or pre-defined. Here all baselines are deployed for the frame-frame tracking problem. Ours first optimizes the feature fitting term based objective (5.4) to get the coarse motion and then refine the graph with the classic point-to-plane constraints using the raw depth maps.

Quantitative Results The quantitative results on the ScanNet dataset and the non-rigid dataset can be found in Table 1. The estimated motions are evaluated using the 3D End-Point-Error (EPE) metric. On ScanNet, Ours(SN) achieves overall better performance than the other N-ICP baselines, especially when the motions are large (*e.g.*on 0→8 and 0→16 frame jump). Note that the ScanNet pre-trained model Ours(SN) even achieves better results than the classic N-ICPs on the non-rigid dataset, indicating a good generalization ability of the learned non-rigid feature, which makes sense considering that the learnable CNN model focuses only on the feature extraction part, and the using of classic optimizer disentangle the direct mapping from images to motion. It also proves the assumption that the rigid and non-rigid surfaces lie in quite close domains. The fine-tuned model Ours(SN+NR) on the non-rigid dataset further improved these numbers.

Qualitative Results Fig. 4.9, Fig. 4.10, Fig. 4.11, and Fig. 4.12 shows the frame-frame tracking results on the non-rigid frame pairs. We selected the frame pairs with relatively large non-rigid motions. N-ICP-1 (DynamicFusion) and N-ICP-2 (VolumeDeform) have trouble dealing with these motions and converged to bad local minimums. Our method manages to converge to the global solutions on these challenging cases. For instance, the ***clothes*** scene

Method	Energy Terms					ScanNet (SN)				Non-Rigid Dataset (NR)			
	ARAP	P2Plane	P2Point	Color	Feature	3D EPE (cm) on ↓ validation/test on frame jump				3D EPE (cm) on ↓ validation/test on frame jump			
						0→2	0→4	0→8	0→16	0→2	0→4	0→8	0→16
N-ICP-0	✓			✓		3.21/3.65	5.03/5.68	8.17/9.66	15.35/18.65	2.29/2.3	4.08/3.3	7.71/6.3	13.63/12.72
N-ICP-1 [67]	✓	✓	✓			2.43/2.75	4.50/5.38	8.11/9.09	14.62/17.10	**1.49**/1.53	2.98/2.71	6.61/6.52	11.14/12.07
N-ICP-2 [114]	✓	✓	✓	✓		**2.04**/2.71	3.58/4.47	6.07/7.89	10.36/14.96	1.68/1.70	3.41/2.60	5.50/5.20	11.80/10.59
Ours (SN)	✓	✓			✓	2.10/**2.60**	**3.55/4.39**	**5.28/6.98**	**7.34/10.59**	1.73/1.60	2.77/2.63	4.99/5.08	7.09/8.32
Ours (SN+NR)	✓	✓			✓	–	–	–	–	1.55/**1.34**	**2.25/2.23**	**4.16/4.50**	**6.47/7.59**

Table 4.1 3D End point Error (EPE) on ScanNet and our Non-Rigid dataset. The frame jumps shows the index of the indices of the source and target frame. The number of unkonws in the deformation graph is 1152 ($16 \times 12 \times 6$). Ours (SN): trained on ScanNet [18]. Ours (SN + NR): pretrained on ScanNet and fine-tuned on the Non-Rigid dataset [9].

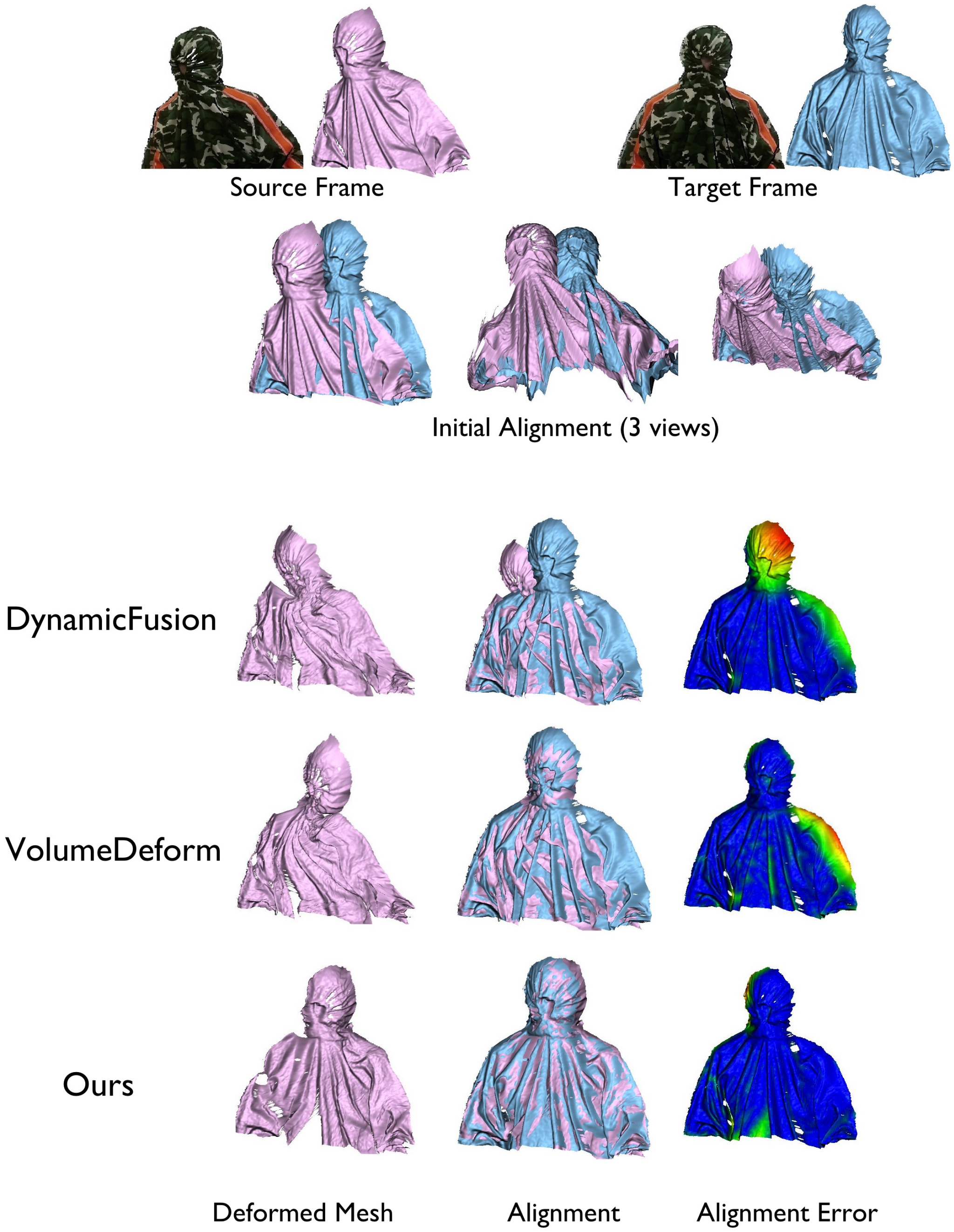


Figure 4.9 Frame-Frame tracking results for the "cloth" sequence. Compared to DynamicFusion Newcombe et al. [67] and VolumeDeform Innmann et al. [41], Our method yield better alignment with target frame. Meshes are constructed from depth images. Depth images are preprocessed by the bilateral filter to reduce observation noise. Initial alignment is done by simply setting the camera poses of both frames to identity. The alignment error (hotter means larger) measures the point to point distance between target mesh and the transformed source mesh.

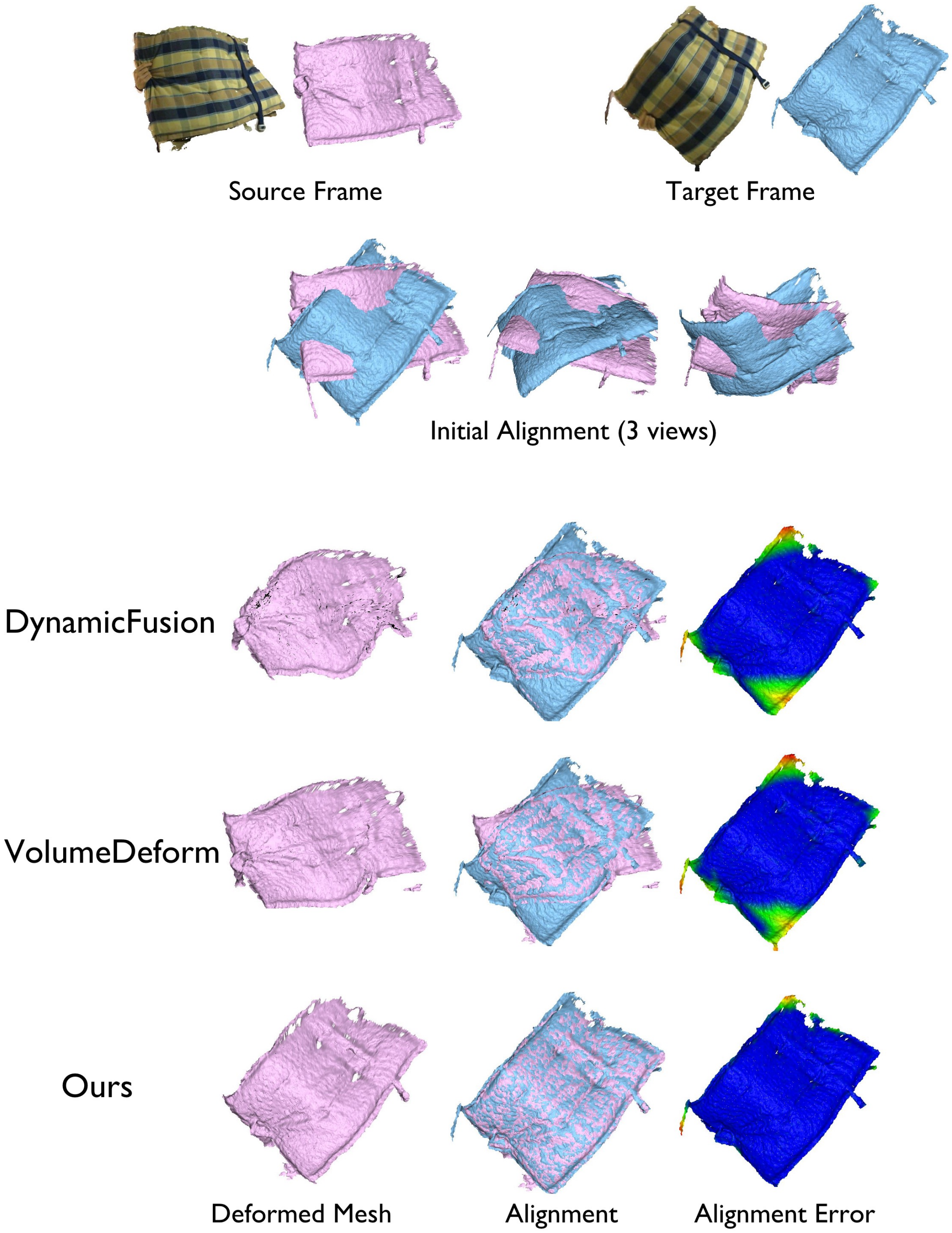


Figure 4.10 Frame-Frame tracking results for the “pillow” sequence. Compared to DynamicFusion Newcombe et al. [67] and VolumeDeform Innmann et al. [41], Our method yield better alignment with target frame. Meshes are constructed from depth images. Depth images are preprocessed by the bilateral filter to reduce observation noise. Initial alignment is done by simply setting the camera poses of both frames to identity. The alignment error (hotter means larger) measures the point to point distance between target mesh and the transformed source mesh.

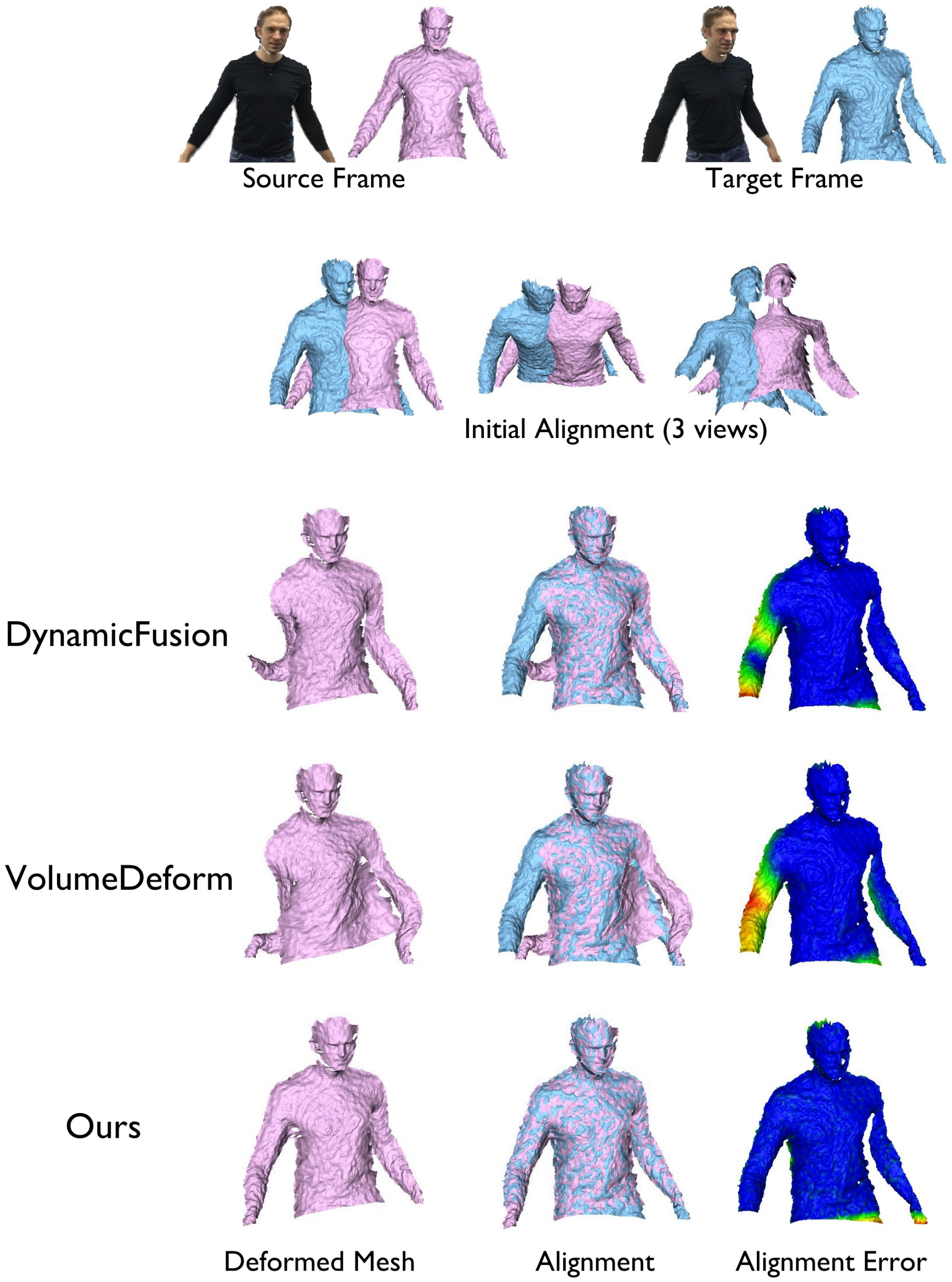


Figure 4.11 Frame-Frame tracking results for the “Adult” sequence. Compared to DynamicFusion Newcombe et al. [67] and VolumeDeform Innmann et al. [41], Our method yield better alignment with target frame. Meshes are constructed from depth images. Depth images are preprocessed by the bilateral filter to reduce observation noise. Initial alignment is done by simply setting the camera poses of both frames to identity. The alignment error (hotter means larger) measures the point to point distance between target mesh and the transformed source mesh.

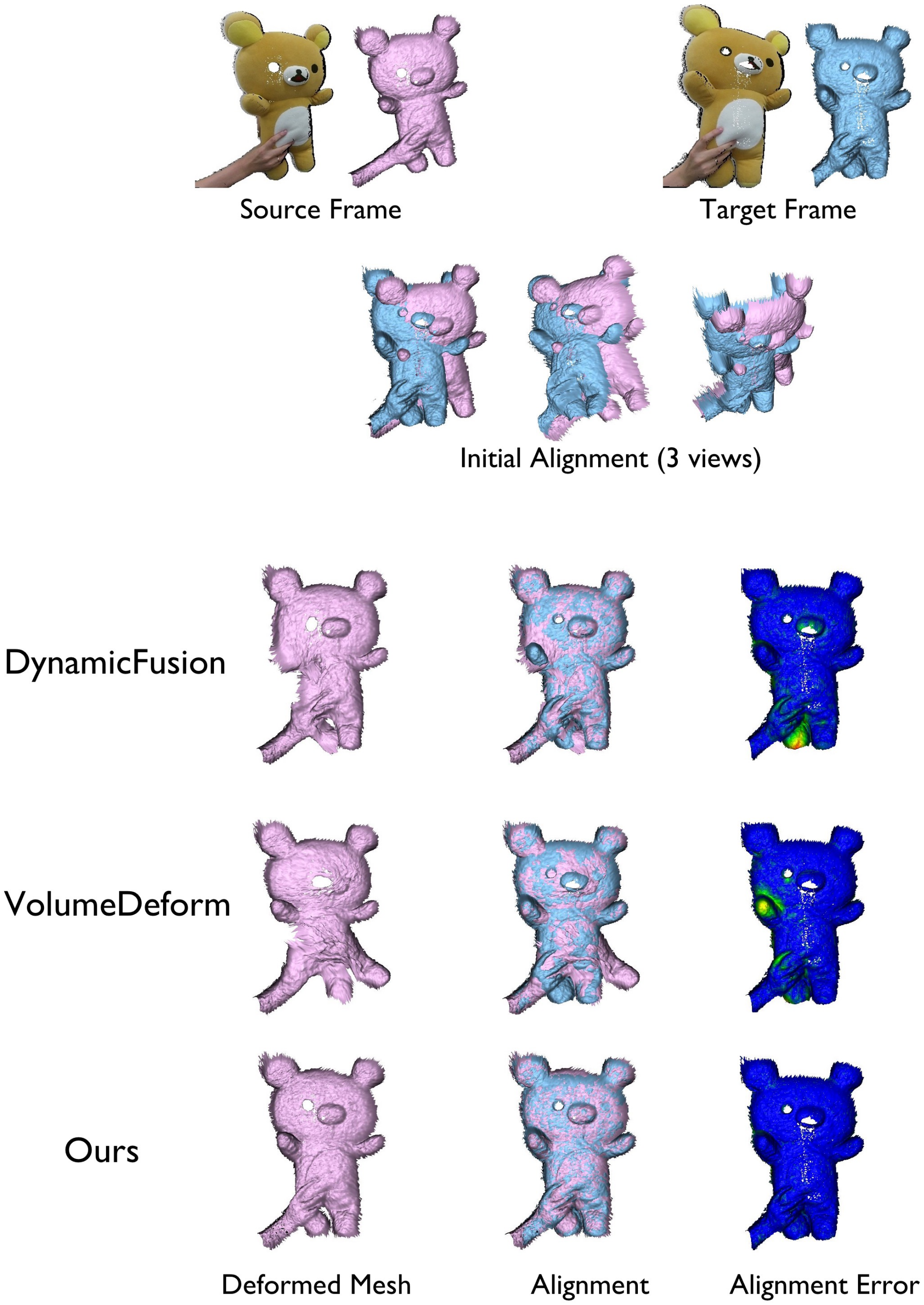


Figure 4.12 Frame-Frame tracking results for the "Bear" sequence. Compared to DynamicFusion Newcombe et al. [67] and VolumeDeform Innmann et al. [41], Our method yield better alignment with target frame. Meshes are constructed from depth images. Depth images are preprocessed by the bilateral filter to reduce observation noise. Initial alignment is done by simply setting the camera poses of both frames to identity. The alignment error (hotter means larger) measures the point to point distance between target mesh and the transformed source mesh.

in Fig. 4.9 is an especially challenging case for classic non-rigid ICP methods because the point-to-plane term has no chance to slide over the zigzag clothes surface which contains multiple folds, and the color consistency term could also be easily confused by the repetitive camouflage textures of the ***clothes***. The learned features show an advantage for capturing high order deformation on those cases. For additional tracking results, please refer to the supplementary material.

4.3.2 Preconditioning results

We randomly collected 10K [**A**, **b**] pairs from different iterations the Gauss-Newton step. We split them to train/valid/test according to the ratio of 8/1/1. We compare with 3 PCG baselines: w/o preconditioner, the standard block-diagonal preconditioner, and the Incomplete Cholesky factorization based preconditioner. We also show the ablation studies on three of the ConditionNet variants: Diagonal, Sparse and Dense. Fig. 4.13 shows the PCG steps using different preconditioners. The learned preconditioner outperforms the classic ones by a large margin. Tab. 2 shows PCG's solving results using different preconditioners. All learned preconditioners significantly reduced the condition numbers. ConditionNet-Dense achieves the best convergence rate and the least overall solving time.

4.4 Summary

In this chapter, we present an end-to-end learning approach for non-rigid RGB-D tracking. Our core contribution is the learnable optimization approach which improves both robustness and convergence by a significant margin. The experimental results show that the learned non-rigid feature significantly improves the convergence of Gauss-Newton solver for the frame-frame non-rigid tracking. In addition, our method increases the PCG solver's convergence rate by predicting a good preconditionier. Overall, the learned preconditioner requires 2 to 3 times fewer iterations until convergence.

4.5 Limitation

While we believe this results are very promising and can lead to significant practical improvements in non-rigid tracking and reconstruction frameworks, there are several major challenges are yet to be addressed:

1) The proposed non-rigid feature extractor adopted plain 2D convolution kernels, which are potentially not the best option to handle 3D scene occlusions. One possible research

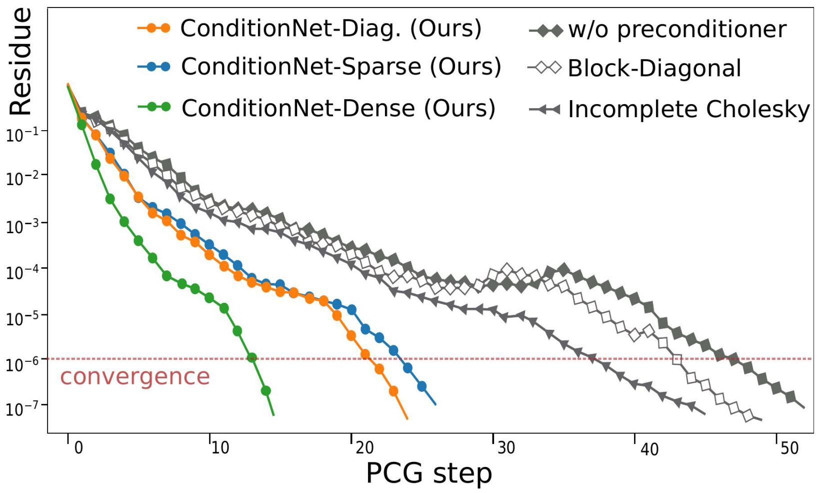


Figure 4.13 PCG convergence using different preconditioners. The curves show the average convergence on the testing dataset. Note that our final method (green curve) requires 3 times fewer PCG steps to achieve the same residual (10^{-6}) than the best baseline (dashed line).

Preconditioner	*density*	κ	*iters*	*time (ms)*
None	–	3442.18	46	33.43
Block-Diagonal	0.46%	541.52	44	31.34
Incomplete Cholesky	1.52	379.82	37	28.42
ConditionNet-Diagonal (ours)	0.46%	93.55	21	12.38
ConditionNet-Sparse (ours)	1.52%	125.81	23	17.80
ConditionNet-Dense (ours)	100.%	**34.90**	**13**	**10.32**

Table 4.2 PCG solving results using different preconditioners (residue threshold of convergence: 10^{-6}). *density*: density of preconditioner. κ: condition number of the modified linear system. *iters*: total steps for convergence. *time(ms)*: time of solving. All numbers are obtained with Pytorch-GPU implementation.

avenue is to directly extract non-rigid features from 3D point clouds or mesh structures using the point-based architectures [75], or even graph convolutions [11].

2) Collecting dense scene flow using DynamicFusion for real-world RGB-D video sequence is expensive (*i.e.* segmentation and outlier removal can become painful processes). The potential solution is learning on synthetic datasets. (*e.g.*using graphics simulations where the dense motion ground truth is available).

3) The proposed method is essentially based on the direct Lucas-Kanade [2] alignment method that uses the gradient on the 2D image/feature grid. If the source and target frames are too far away from each other (e.g. two frames have no overlap at all), the direct alignment method will not work. Fig. 4.14 shows a large and a small overlap case. A potential solution in such situation is to use the sparse descriptor matching [9, 59, 4] method to coarsely draw two frames together and then use the direct method to refine the tracking.

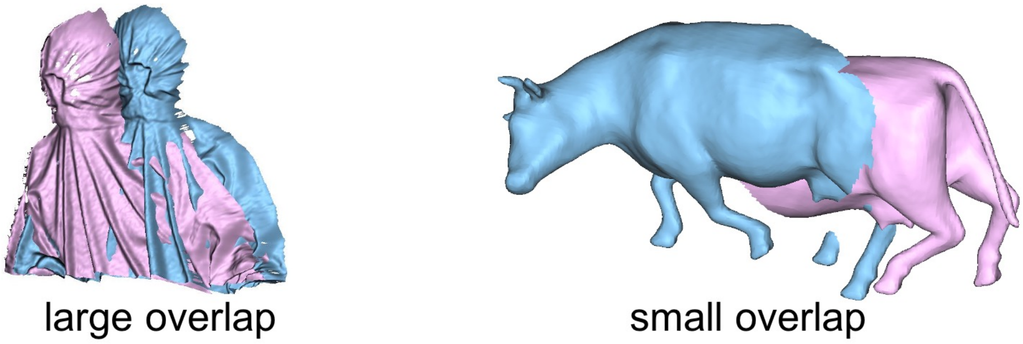


Figure 4.14 Left: frames pair with large overlap (most points in the source frame are observed in the target frame); Right: frame pair with small overlap (only the points around the stomach of the cow are observed in both frames). To handle small overlap cases, it is necessary to learn the overlapping area between two frames.

4.6 Supplementary material

In this document, we provide more details to the main chapter. In Sec. 4.6.1, we show additional non-rigid tracking results. In Sec. 4.6.2, we provide details of about our networks. In Sec. 4.6.3, we provide details of about of the non-rigid dataset. Alg. 1 and Alg. 2 provide details about the nested optimizers: Gauss-Newton (GN) and Preconditioned Conjugate Gradient (PCG).

4.6.1 Additional non-rigid tracking results

Fig. 4.16 and Fig. 4.15 show the frame-frame non-rigid tracking results on different ranges of motion, from small to large. We compare our method with two baselines, DynamicFusion [67] and VolumeDeform [114]. Though all three methods show artifacts when the motion is large, the results of our method are geometrically closer to the target frames.

4.6.2 Network configurations

Table.4.3 shows the network structure of the non-rigid feature extractor. It is based on the fully convolutional networks [57]. The RGB-D images are resized to 128×96 before feeding to the feature extractor. All convolutions are followed by a batch normalization layer and a ReLU layer. We use spatial average pooling of size 2 to down-sample features between two feature pyramids. The input volume is [128, 96, 4], where 4 represents the 4 channels of the RGB-D frame. Down-sampling is applied three times in total. The output volume of the network is [16, 12, 5]. Table. 4.4 shows the network structure of the ConditionNet. The number of input and output channels for ConditionNet-Dense and ConditionNet-Sparse is 1. ConditionNet-Diagonal has 36 channels for input and 21 channels for output.

4.6.3 Dataset details

Fig. 4.17 shows the examples in the collected non-rigid dataset. Foreground are selected by crowd-sourced manual annotation. Similar to the ScanNet, the RGB-D videos are captured via iPad mounted Structure Sensor. The depth frames are recorded at a resolution of 640×480. A variety of dynamic objects are included: animals, adults, children, cloth, furniture, *etc*. Fig 4.18 shows the statistics of the dataset. The backgrounds vary from static scenes to dynamic ones with multiple moving objects. Fig. 4.19 and Fig. 4.20 show the Examples of reconstructed model using our non-rigid tracking and reconstruction tool, from which we get the dense motion annotation for network training.

Algorithm 1 Gauss-Newton Optimization

```
$\mathscr{C} \leftarrow \Phi(\mathscr{I}_s, \mathscr{I}_t)$    ▷ Estimate correspondences
$\mathscr{W} \leftarrow \Psi(\mathscr{Z}_s, \mathscr{Z}_t', \mathscr{H})$    ▷ Estimate importance weights
function SOLVER($\mathscr{C}, \mathscr{W}, \mathscr{V}$)
    $\mathscr{T} \leftarrow \mathbf{0}$
    for $n \leftarrow 0$ to $max\_iter$ do
        $\mathbf{J}, \mathbf{r} \leftarrow$ ComputeJacobianAndResidual($\mathscr{V}, \mathscr{T}, \mathscr{Z}_\mathrm{s}, \mathscr{Z}_\mathrm{t}', \mathscr{C}, \mathscr{W}$)
        $\Delta\mathscr{T} \leftarrow$ PreconditionedConjugateGradient($\mathbf{J}^T\mathbf{J}\Delta\mathscr{T} = -\mathbf{J}^T\mathbf{r}$)    ▷ Solve linear system
        $\mathscr{T} \leftarrow \mathscr{T} + \Delta\mathscr{T}$    ▷ Apply increment
    return $\mathscr{T}$
```

Algorithm 2 Preconditioned Conjugate Gradients, reproduced from [3]

Input:

$A(\cdot)$ // A function which implements $A\mathbf{x}$

$\mathbf{b}$ // The $\mathbf{b}$ vector in the linear system

$\mathbf{x}$ // The initial value of the state $\mathbf{x}$

$M^{-1}(\cdot)$ // A function which implements a preconditioner

n // the number of iterations

Output:

$\mathbf{x}$ // $\mathbf{x}$ such that $A(\mathbf{x}) \approx \mathbf{b}$

```
$i \leftarrow 0$
$\mathbf{r} \leftarrow \mathbf{b} - A(\mathbf{x})$
$\mathbf{d} \leftarrow M^{-1}(\mathbf{r})$
$\lambda_{new} \leftarrow \mathbf{r}^\mathrm{T}\mathbf{d}$
while $i < n$ do
    $\mathbf{q} \leftarrow A(\mathbf{d})$
    $\alpha \leftarrow \frac{\lambda_{new}}{\mathbf{d}^\mathrm{T}\mathbf{q}}$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{d}$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q}$
    $\mathbf{s} \leftarrow M^{-1}(\mathbf{r})$
    $\lambda_{old} \leftarrow \lambda_{new}$
    $\lambda_{new} \leftarrow \mathbf{r}^\mathrm{T}\mathbf{s}$
    $\beta \leftarrow \frac{\lambda_{new}}{\lambda_{old}}$
    $\mathbf{d} \leftarrow \mathbf{s} + \beta\mathbf{d}$
    $i \leftarrow i + 1$
```

Module Name	Channels	Stride	Batch Norm.	Activation	Dilation
Conv2d	16	1	True	ReLU	1
Conv2d	32	1	True	ReLU	2
Conv2d	32	1	True	ReLU	2
Average Polling					
Conv2d	32	1	True	ReLU	1
Conv2d	64	1	True	ReLU	2
Conv2d	64	1	True	ReLU	2
Average Polling					
Conv2d	64	1	True	ReLU	1
Conv2d	96	1	True	ReLU	2
Conv2d	96	1	True	ReLU	2
Average Polling					
Conv2d	96	1	True	ReLU	1
Conv2d	128	1	True	ReLU	2
Conv2d	128	1	True	ReLU	2
Conv2d	5	1	True	ReLU	1

Table 4.3 The network configuration of the non-rigid extractor.

Module Name	Channels	Stride	Batch Norm.	Activation	padding
Input	n_{in}				
Conv2D-1.1	32	1	true	PReLU	1
Conv2D-1.2	32	1	true	PReLU	1
Max pooling					
Conv2D-2.1	64	1	true	PReLU	1
Conv2D-2.2	64	1	true	PReLU	1
Max pooling					
Conv2D-3.1	128	1	true	PReLU	1
Conv2D-3.2	128	1	true	PReLU	1
Max pooling					
Conv2D-4.1	128	1	true	PReLU	1
Conv2D-4.2	128	1	true	PReLU	1
NearestUpSample1					
Concat (w/ Conv2d-3.2)	128+128	1	true	PReLU	1
Conv2D	128	1	true	PReLU	1
Conv2D	128	1	true	PReLU	1
NearestUpSample2					
Concat (w/ Conv2d-2.2)	128+64	1	true	PReLU	1
Conv2D	64	1	true	PReLU	1
Conv2D	64	1	true	PReLU	1
NearestUpSample3					
Concat (w/ Conv2d-1.2)	64+32	1		PReLU	1
Conv2D	32	1	true	PReLU	1
Conv2D	n_{out}	1	true	Linear	1

Table 4.4 The network configuration of ConditionNet. $n_{in} = 1, n_{out} = 1$ for ConditionNet-Dense and ConditionNet-Sparse. $n_{in} = 36, n_{out} = 21$ for ConditionNet-Diagonal. The kernel size is 2x2 for ConditionNet-Diagonal and alternate between 1x1 and 2x2 for ConditionNet-Dense and ConditionNet-Sparse.

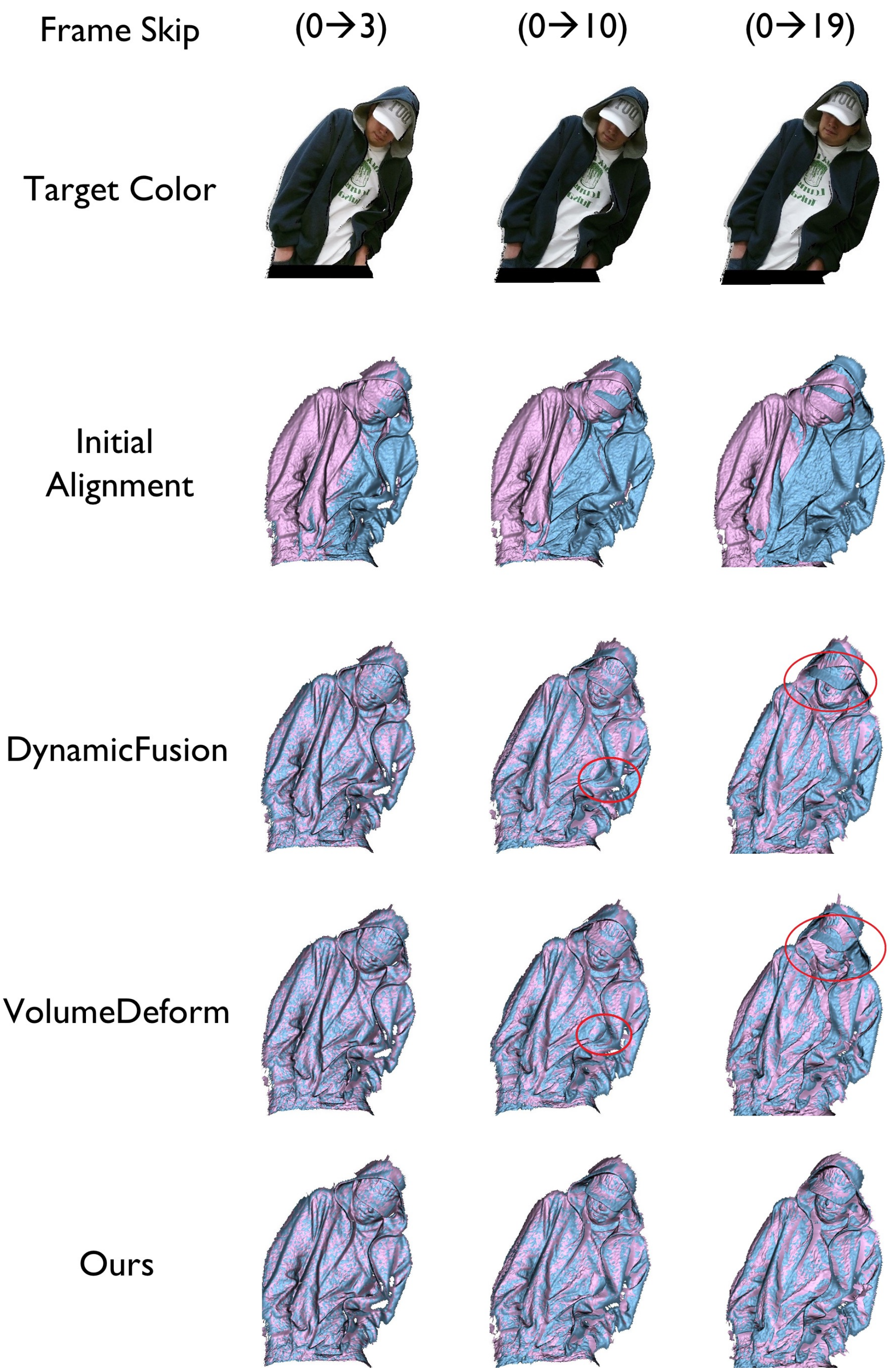


Figure 4.15 Non-Rigid tracking results on the ***hoody*** scene.

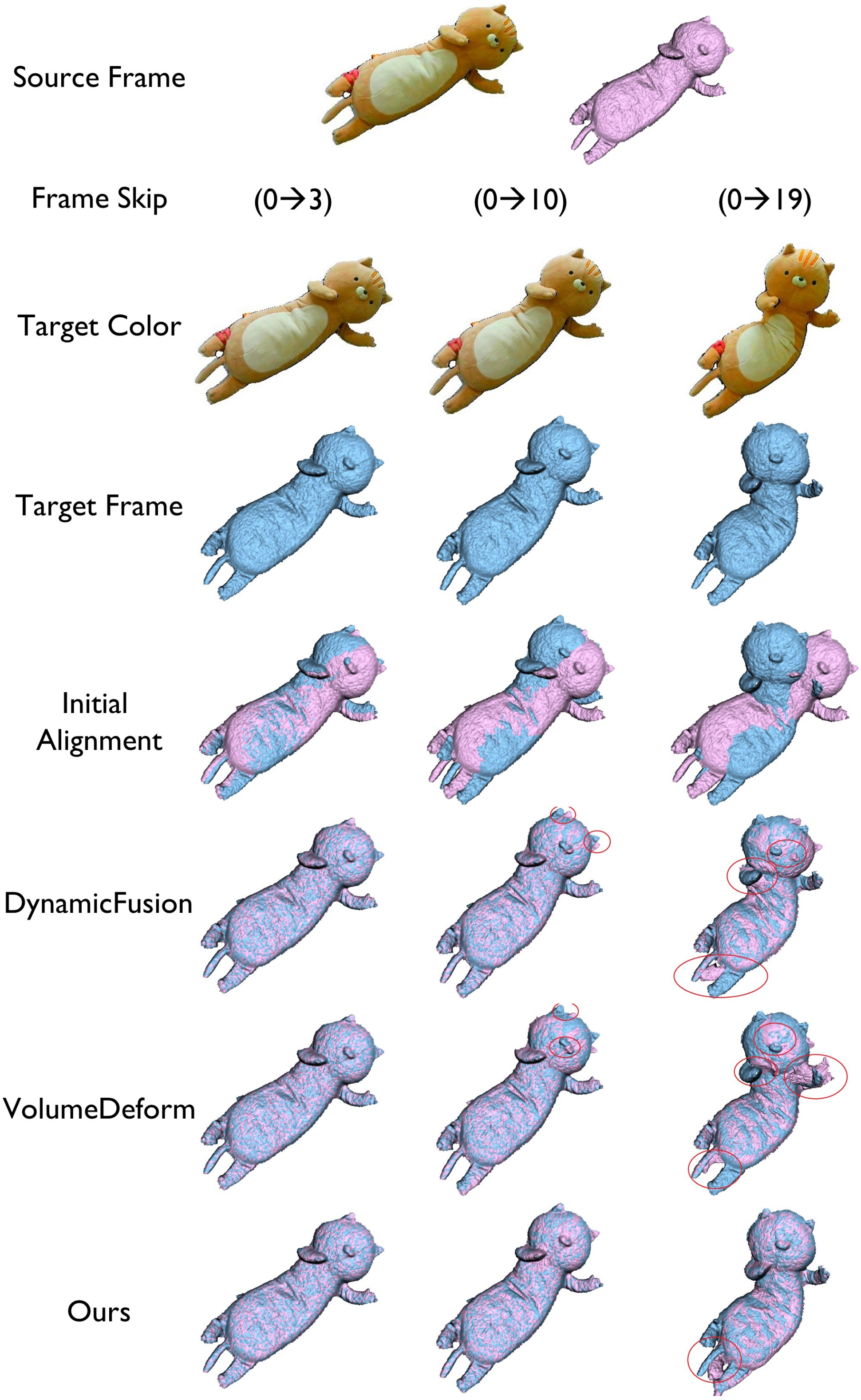


Figure 4.16 Non-Rigid tracking results on the ***kitty*** scene.

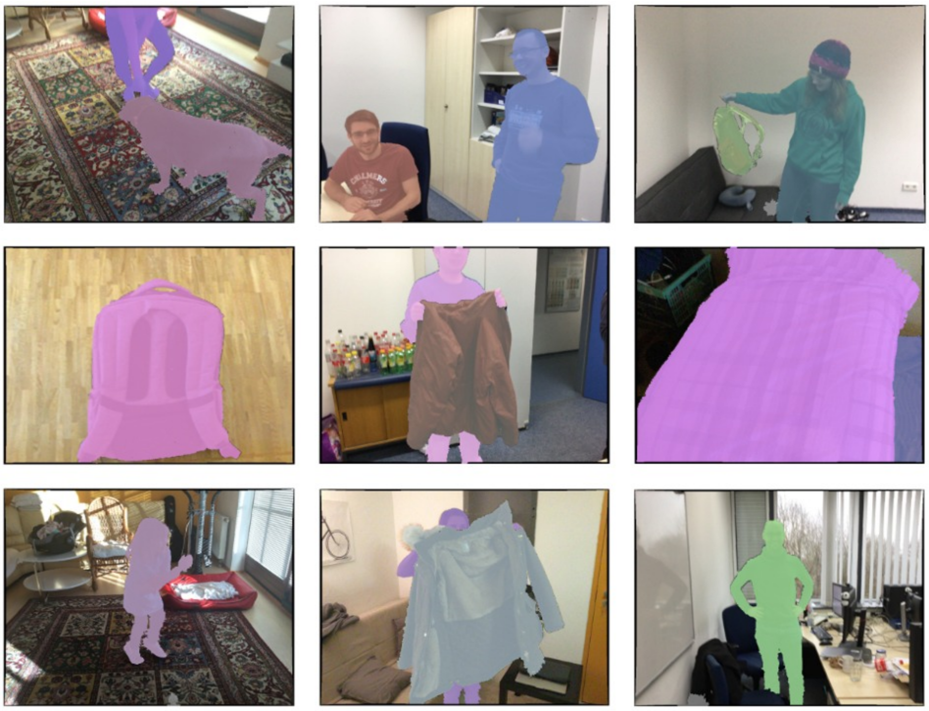


Figure 4.17 The Non-rigid dataset from [9] contains a large variety of dynamic sequences with segmentation masks and point correspondences between different RGB-D frames. The foreground are segmented through crowd-sourced manual annotation.

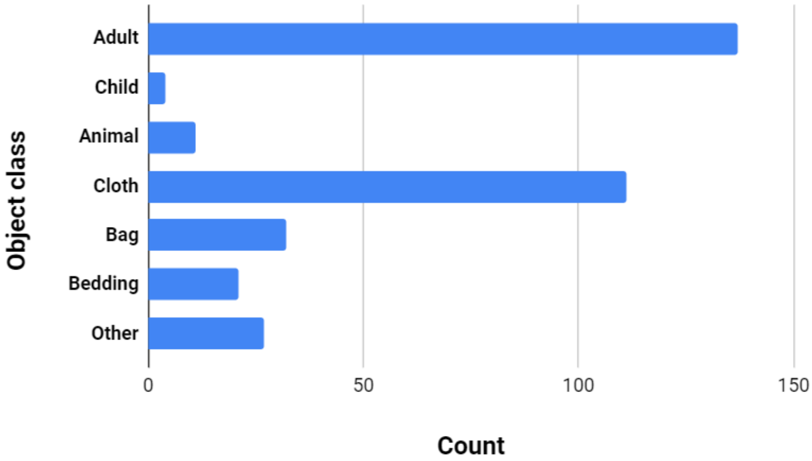


Figure 4.18 Object class variety. This dataset include many different sequences of dynamic objects, such as cloths, bags, etc..

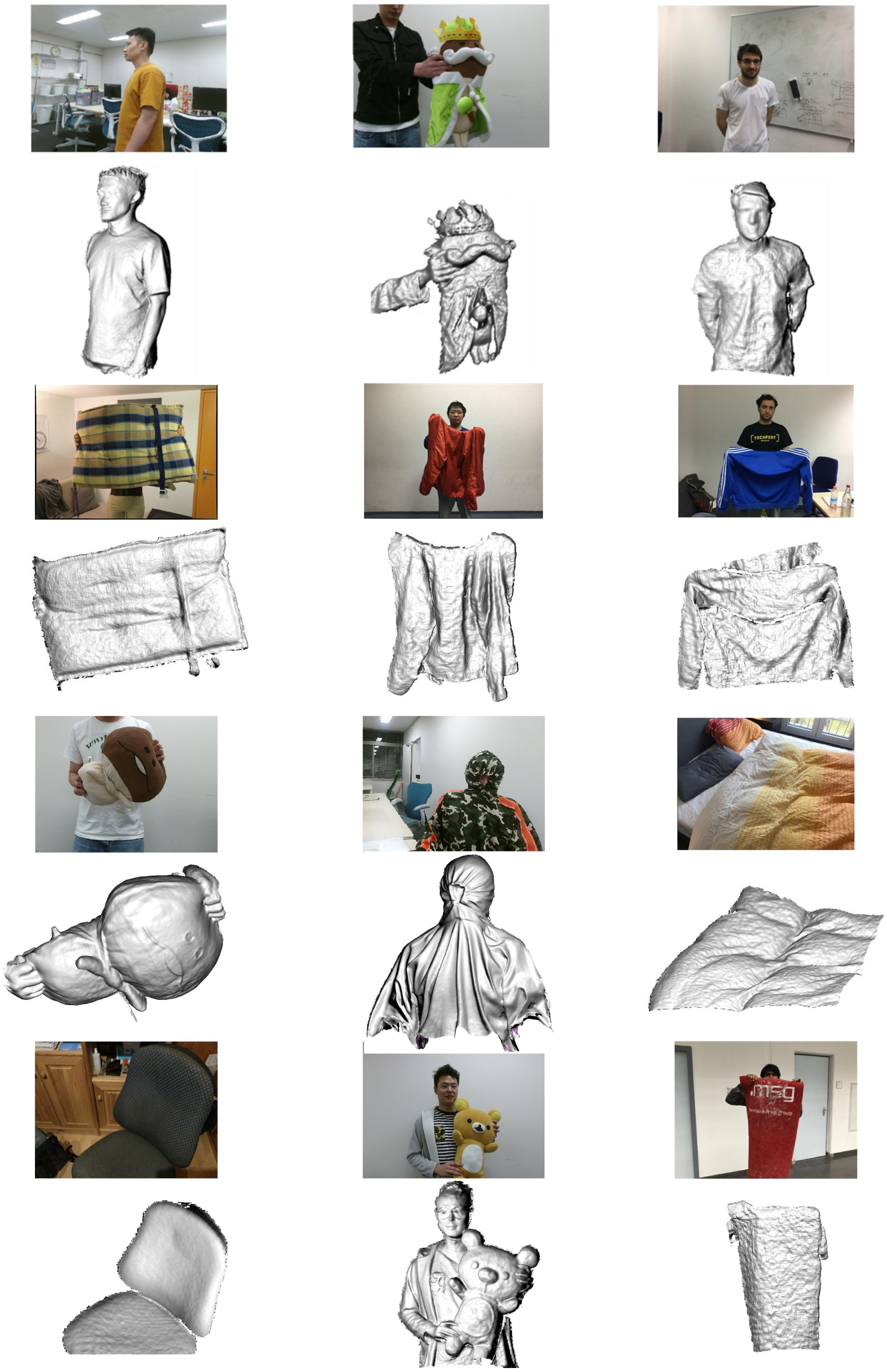


Figure 4.19 Examples of reconstructed model in the non-rigid dataset. Top row: Color frame in the sequence. Bottom row: The reconstructed foreground object models using our non-rigid tracking and reconstruction method.

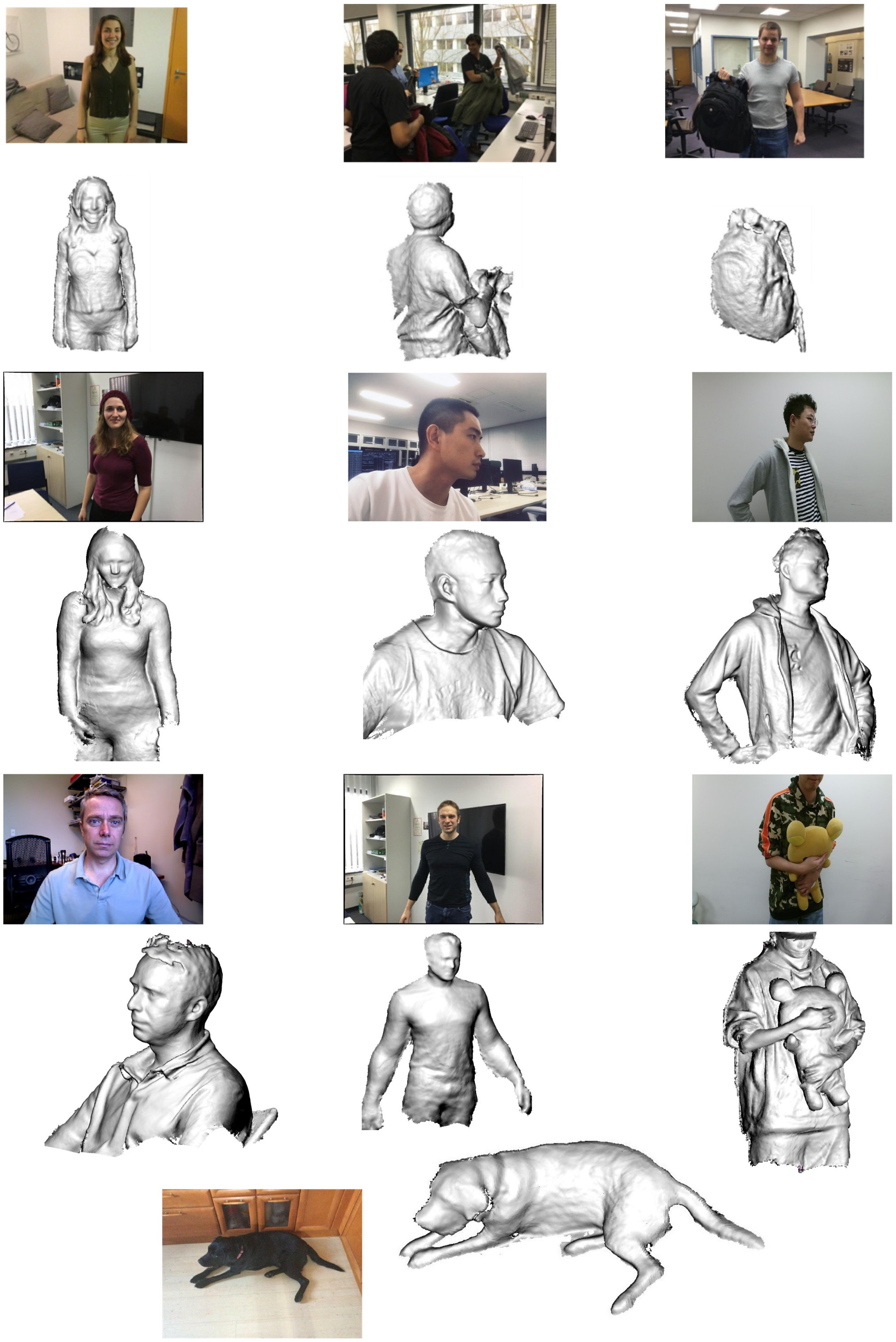


Figure 4.20 Examples of reconstructed model in the non-rigid dataset. Top row: Color frame in the sequence. Bottom row: The reconstructed foreground object models using our non-rigid tracking and reconstruction method.

Chapter 5

SplitFusion

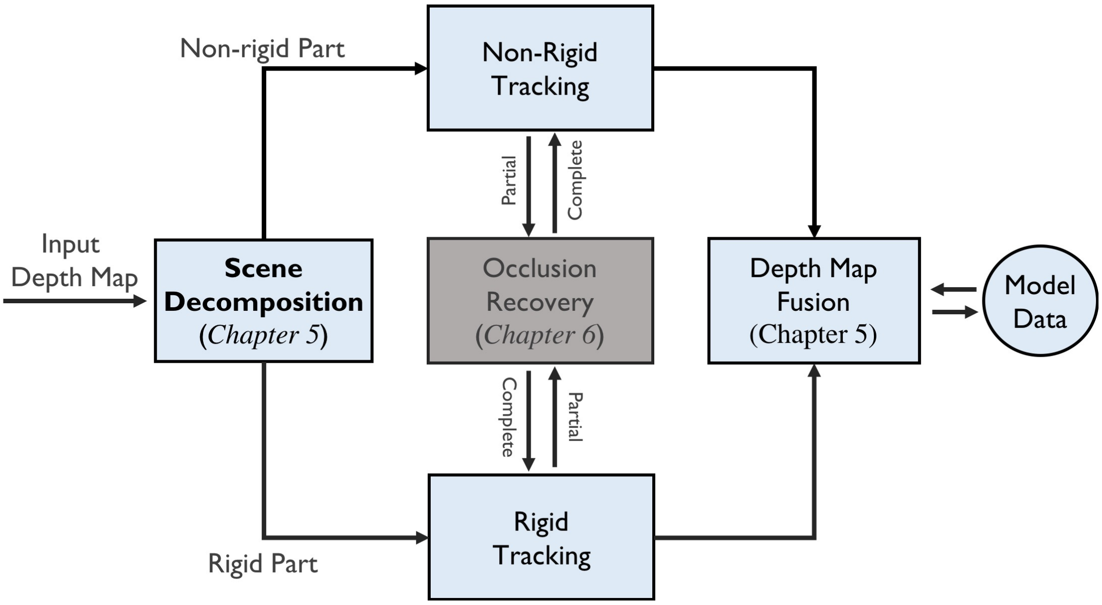


Figure 5.1 The focus of this chapter and its position in the entire system.

As shown in Figure. 5.1, this chapter focuses on the main framework of this system: a framework that allows simultaneous tracking and mapping on both rigid and non-rigid scene. We present SplitFusion, a novel dense RGB-D SLAM framework that simultaneously performs tracking and dense reconstruction for both rigid and non-rigid components of the scene. SplitFusion first adopts deep learning based semantic instant segmentation technique to split the scene into rigid or non-rigid surfaces. The split surfaces are independently tracked via rigid or non-rigid ICP and reconstructed through incremental depth map fusion. Experimental results show that the proposed approach can provide not only accurate environment maps but also well-reconstructed non-rigid targets, *e.g.*the moving humans.

5.1 Introduction

Visual Simultaneous Localization and Mapping (Visual SLAM) is a popular robotics research topic which focuses on the robot self-localization and unknown environment reconstruction using visual sensors, such as stereoscopes, monocular cameras, RGB-D sensors, and laser scanners. Most of the existed visual SLAM approaches are designed based on the static environment assumption. However, dynamic elements in the scene can cause trouble for camera 6-DoF pose tracking, leading to artifacts in the reconstruction.

To deal with the problem of the dynamic environment, the recent works [80] extract the dynamic components from the input, removing them as exceptions to apply static SLAM frameworks. These approaches have shown improvements in both camera tracking and reconstruction. Nonetheless, the removed dynamic objects are important targets for many autonomous robot systems, such as cooperative manipulation and human-robot interactions.

The seminal DynamicFusion [67] proposed a general solution for non-rigid scene reconstruction by parameterizing the scene with a expandable deformation model. However, the complexity of non-rigid tracking grows quadratically with the model's size, making this approach less scalable to larger scenes. Moreover, without explicit surface segmentation, these methods can not efficiently handle topology changes.

In this chapter, we present a novel SLAM framework called SplitFusion which simultaneously performs tracking and reconstruction for both rigid and non-rigid components of the scene. We first split the scene into rigid or non-rigid geometric surfaces by leveraging the recent advancement in deep learning based semantic instance scene understanding. The split surfaces are independently tracked via rigid or non-rigid ICP and reconstructed through incremental depth map fusion. The proposed visual SLAM framework not only results in accurate and clean rigid environment maps but also provides well reconstructed non-rigid objects.

In this chapter, we present a novel non-rigid scene reconstruction system that by leveraging the recent advancement in semantic/instance scene understanding, to simultaneously reconstruct multiple non-rigidly deforming objects along with the static scene together.

5.2 Method

SplitFusion Overview

SplitFusion splits a scene into several mutually independent geometric surfaces. A surface is associated with a warping filed that transforms the surface into the time-varying frames. As show in Fig. 5.1 the followings are the overview of this system:

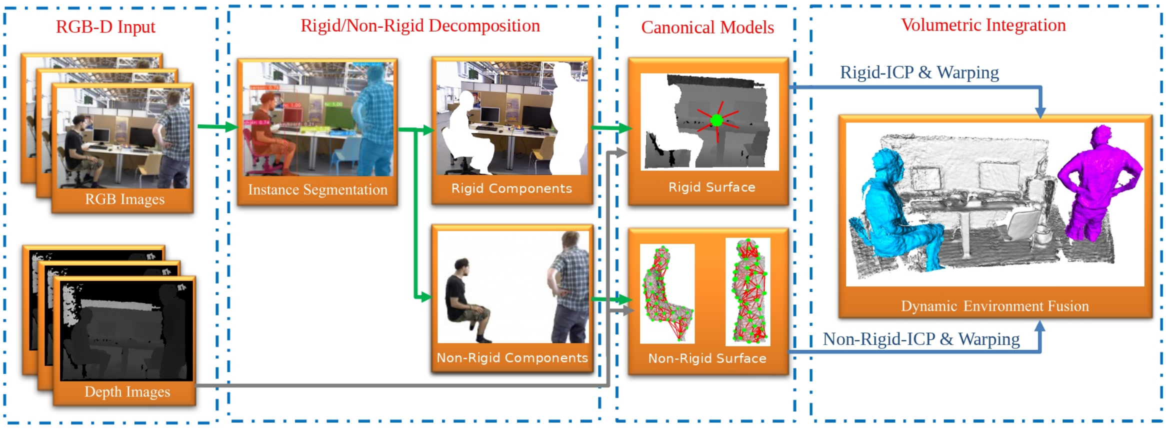


Figure 5.1 Overview of the proposed reconstruction system. For input RGB-D frames, we first apply YOLACT [7] based semantic instance segmentation on the color images and then refine the segmentation via depth map (*i.e.*point clustering). The segmented surfaces are tracked independently. The tracking is performed using rigid-ICP for rigid classes, (e.g. chairs and ground) or non-rigid ICP for deformable classes (e.g. human and animals). Depth map fusion is done for each surface and the extracted meshes are combined together using the relative surface-to-camera transformations.

1. The scene is first decomposed into several independent sub-surfaces. (cf. Section. 5.2.1)
2. The motion of a sub-surface is approximated by a dense warping filed. (cf. Section. 5.2.2)
3. Non-linear optimization is performed to find the best warping filed parameters for each sub-surface. (cf. Section. 5.2.3)
4. A live frame is fused into the warped model using volumetric method. (cf. Section. 5.2.4)

5.2.1 Scene decomposition

SplitFusion takes RGB-D frames as input. Each frame is decomposed into several pieces via deep learning based semantic/instance segmentation and the segmentation are then refined by geometric post-processing. The system flowchart of the proposed method is shown in Fig. 5.1. The RGB frame is first fed into YOLACT [7] to detect the object category and perform pixel-level segmentation. YOLACT is a real-time instance segmentation method that can handle 80 types of objects in 2D images. Unlike two-stage methods such as Mask-RCNN, YOLACT is a one-shot instance segmentation method, which greatly reduces the inference time. It divide the instance segmentation into two parallel tasks. The first task uses fully convolutional networks [57] to generate a set of prototype masks with the same size for each image, and the output uses ReLU function for non-linearization. The second task is object detection based on anchor. It contains three branches: the first branch is used to predict mask coefficients for each prototype, the second branch is used to predict the confidence of instance categories and the third branch is used to predict the coordinates of the bounding box.

There are over 80-type of objects that can be detected by YOLACT. The objects in the environments are classified into rigid or nor-rigid objects according to their semantic labels. For instance, we treat humans and animals as non-rigid, tables, grounds, and desktops as rigid. The next step is to split the 3D surface according to the former 2D segmentation.

However, the YOLACT based segmentation is noisy and there are inevitable misalignments between color image and depth maps of a commodity RGB-D camera. Hence, we refine the YOLACT segmentation based on depth-map based geometry post-processing. This is done by applying the point cloud clustering as demonstrated in [108]. Specifically, the segmented non-rigid masks are first projected to 3D space using the pinhole camera model, then these projected points are used as foreground prior to start a 3D graph-cut [33] based point clouds splitting. It treats every 3D point as a vertex and the vertices are connected with their neighbors by edges. Given these mask points as foreground priors, it split the non-rigid object point clouds out from the rigid backgrounds by computing the weights of the edges.

5.2.2 Per surface warping filed

The decomposed surfaces are processed independently. Similar to [67][41], we represent the per-surface warping filed by the deformation graph $\mathscr{G}$. In the deformation graph, the node i's motion is parameterized by a translation vector $\mathrm{t}_i \in \mathbb{R}^3$ and a rotation matrix $\mathrm{R}_i \in \mathrm{SO3}$. Therefore a node is parameterized by a rigid 6-dof transformation. Putting all parameters into a single vector, we get the warping filed parameters $\mathscr{G} = \{\mathrm{t}_i, \mathrm{R}_i | i = 1...k\}$, where k is the total number of node. Besides the 6-dof motion, a node also has the attribute $\mathrm{g}_i \in \mathbb{R}^3$, storing its 3D position in each time-step. The deformation nodes are sampled in the way that it covers the surface evenly. The nodes are connected based on closeness. Refer to [88] for the details.

The motion of the entire surface is estimated by *linearly blending* [88] the motions from the graph nodes. The influence of a rigid transformation is centered at a node's position, so

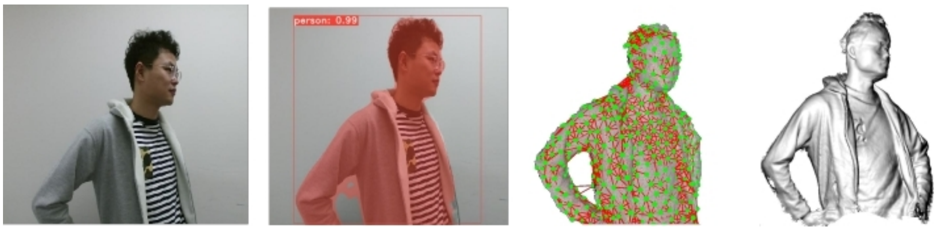


Figure 5.2 Non-Rigid surface representation. From left to right, input RGB image, semantic instant segmentation, deformation graph and surface.

that any nearby point p is mapped to position p̃ according to

$$\tilde{\mathrm{p}} = \mathrm{R}_i(\mathrm{p} - \mathrm{g}_i) + \mathrm{t}_i + \mathrm{g}_i \tag{5.1}$$

The influence of multiple graph node can be smoothly blended so that the transformation in point $\tilde{\mathrm{v}}_j$is a weighted sum of the deformation graph transformations

$$\tilde{\mathrm{v}}_j = \sum_{i=1}^{k} w_{i,j}[\mathrm{R}_i(\mathrm{v}_j - \mathrm{g}_i) + \mathrm{t}_i + \mathrm{g}_i] \tag{5.2}$$

The blending weights $w_{i,j}$ are pre-computed acoording to

$$w_{i,j} = (1 - ||\mathrm{v}_i - \mathrm{g}_j||/d_{max})^2 \tag{5.3}$$

and then normalized to sum to one. Here d_{max} define the max distance of a valid neighbor. The blending weights for nodes that are beyond d_{max} are set to zero. d_{max} is computed per-point. It is defined by the distance to the $K+1$-th nearest neighbor node, i.e. only K nearest neighbor nodes can affect a point. Empirically we use $K = 6$ in our implementation.

Based on the semantic information obtained from YOLACT, we split the scene into independent rigid and non-rigid surfaces. Note that the rigid surface can be considered as a special case of the non-rigid surface, where there are only one node in the deformation graph, *i.e.*, it has only 6 degrees of freedom.

In the rigid case, the non-rigid surface tracking task is reduced to the 6-dof camera pose tracking and the d_{max} is set to positive infinity, *i.e.*all linear blending weights in Eqn. 5.2 become 1.

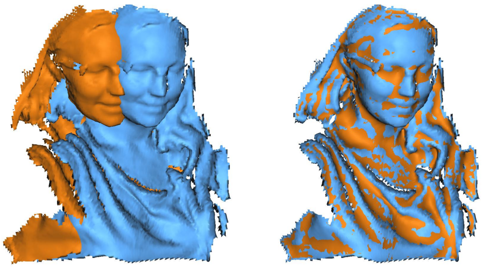


Figure 5.3 Estimation of warping filed parameters between to surfaces using the non-rigid ICP method. Brown: live frame, Blue: canonical surface. After performing non-rigid ICP, the source surface is warped to the target surface via the estimated dense warping filed. the registered surfaces are then fused together using the volumetric method.

5.2.3 Warping filed estimation

We solve the surface motion tracking problem via the non-rigid ICP approach. For each surface, we estimate the deformation filed $\mathcal{G}$ given a target depth map by minimizing the following energy function:

$$\mathrm{E}_{total}(\mathcal{G}) = \mathrm{E}_{data}(\mathcal{G}) + \lambda \mathrm{E}_{prior}(\mathcal{G}) \tag{5.4}$$

Here the data term E_{data} is the dense model-to-frame ICP cost measuring the point-to-plane term between the warped model, which is a projected depth map $\mathrm{D_m}$, and live depth map $\mathrm{D_t}$. It is defined as

$$\mathrm{E}_{data}(\mathcal{G}) = \sum_{(u_m,u_t)\in\Omega} ||n_m^T(v_m - v_t)||^2 \tag{5.5}$$

where Ω is the set of corresponding pixel pairs in the projected depth map and live depth map. $n, v \in \mathbb{R}^3$ are the normal vector and re-projected 3D vertex associated with a pixel. The method to found the correspondence set Ω can be found in the literature[67][41]. The point-to-plane data-term interact with the motion filed parameters according to the chain rule.

Here we show how to compute the Jacobian matrix J for the motion filed parameter $\mathcal{G} = \{\mathrm{t}_i, \mathrm{R}_i | i = 1...k\}$ w.r.t the point-to-plane term. According the to Gauss-Newton method, the point-to-plane residual r is computed as:

$$r = n_m^T(v_m - v_t)$$

Here, only v_m is relevant, n_m^T and v_t are constant values. Accordingly,

$$\frac{\partial r}{\partial v_m} = n_m^T$$

$$J = \frac{\partial r}{\partial \mathscr{G}} = \{\frac{\partial r}{\partial \mathrm{t}_i}, \frac{\partial r}{\partial \mathrm{R}_i} | i = 1...k\}$$

$$\frac{\partial r}{\partial \mathrm{t}_i} = \frac{\partial r}{\partial v_m}\frac{\partial v_m}{\partial \mathrm{t}_i}$$

$$\frac{\partial r}{\partial \mathrm{R}_i} = \frac{\partial r}{\partial v_m}\frac{\partial v_m}{\partial \mathrm{R}_i}$$

The partial derivatives $\partial v_m/\partial \mathrm{t}_i$ and $\partial v_m/\partial \mathrm{R}_i$ can be computed via the linear blending function defined in Eqn. 5.2.

Following the engineering insights from [67, 41], we also use the point-to-point constraint at the beginning iterations of N-ICP to fast close-up the gap between two geometries and use point-to-plane later for alignment refinement.

The prior term E_{prior} regularize the shape deformation. We use the ARAP [85], which encourages locally rigid motions. It is defined as

$$\mathrm{E}_{prior}(\mathscr{G}) = \sum_{i=1}^{k} \sum_{j \in \mathscr{N}_i} \mathscr{E}_{i,j} \cdot ||(\mathrm{t}_i - \mathrm{t}_j) - \mathrm{R}_i(\mathrm{t}_i^{'} - \mathrm{t}_j^{'})||^2 \tag{5.6}$$

Where $\mathscr{N}_i$ denotes node-i's neighboring nodes, and $\mathrm{t}_j^{'}$, $\mathrm{t}_j^{'}$ are the positions of i, j after the transformation. $\mathscr{E}_{i,j}$ define the weight associated with the edge.

The energy $\mathrm{E}_{total}(\mathscr{G})$ is then optimized by the Gauss-Newton update steps:

$$(\mathrm{J}^{\mathrm{T}}\mathrm{J})\Delta\mathscr{G} = \mathrm{J}^{\mathrm{T}}\mathrm{r} \tag{5.7}$$

where r is the error residue, and J is the Jacobian of the residue with respect to $\mathscr{G}$. This linear equation is solved using the data-parallel preconditioned conjugate gradient (PCG). Note that by splitting the whole scene into sub-surfaces, we also decomposed a potentially very large linear system into several smaller ones, which are easier to solve. If the surface is rigid, the tracking is reduced to rigid point-to-plane ICP and the ARAP regularization disappeared from the energy function since there is only one deformation node.

5.2.4 Surface fusion

We use the TSDF function to update each sub-surface model geometry. Following the fusion technique introduced by [66][67], the depth maps segments D_t of the real-time RGB-D frame is incrementally integrated into the canonical TSDF. Note that each sub-surface is associated with a separate volume and the TSDF fusion is also performed independently. We may have both rigid and non-rigid surfaces. Non-rigid surface fusion is a generalization of the projective truncated signed distance function integration approach applied in the rigid case in [66].

After the dense TSDF volumes are created, we perform per pixel ray-casting [71] to extract the final reconstructed surfaces. All surfaces are re-united to the camera coordinate system according to the surface-to-camera warping fileds (cf. Section 5.2.3).

5.3 Experimental results

We evaluate the proposed approach in two perspectives: non-rigid scene reconstruction and camera pose tracking w.r.t the rigid background.

5.3.1 Non-rigid scene reconstruction results

We evaluate the reconstruction results of our method using the following three published non-rigid sequences:

freiburg_walking_xyz This sequence is from the TUM-RGBD [87] dataset. This scene contains two walking people in an office. It is captured via the Asus Xtion sensor which has manually been moved along three directions (xyz) while keeping the same orientation. This sequence is extremely challenging because the non-rigid human motions are very fast and the people are also interacting with the rigid objects, *i.e.* the chair. The sensor readings for the chairs are insufficient for detection, leading to environmental noise for tracking. We compare our method with the following methods: the rigid SLAM: KinectFusion [66], the non-rigid reconstructor: DynamicFusion [67], and Co-Fusion [77] which also perform rigid tracking for segmented objects. Reconstruction results for this sequence can be found in Fig. 5.4 and 5.5. KinectFusion does not work for dynamic scenes. DynamicFusion treats the entire scene as a whole non-rigid object, the scene sticks together over time. Although Co-Fusion also uses segmentation technique, it can not track and reconstruct the non-rigid human. We achieve significantly better reconstructions than these methods. For video comparison, we refer to the supplementary material.

Selfie This sequence is released by StaticFusion [80]. In this sequence, a person carries the camera with her right arm while the camera points at them. This a very complex test because the person often occupies more than 50% of the image and which causes a problem for camera pose tracking and background reconstruction. The result of this sequence is shown in Fig. 5.6. We achieve robust tracking and reconstruction for both the foreground people and

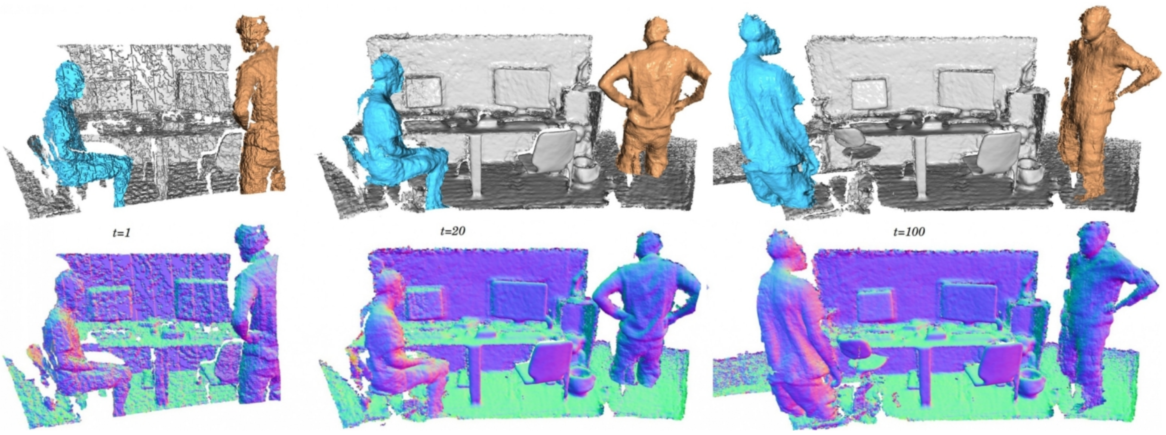
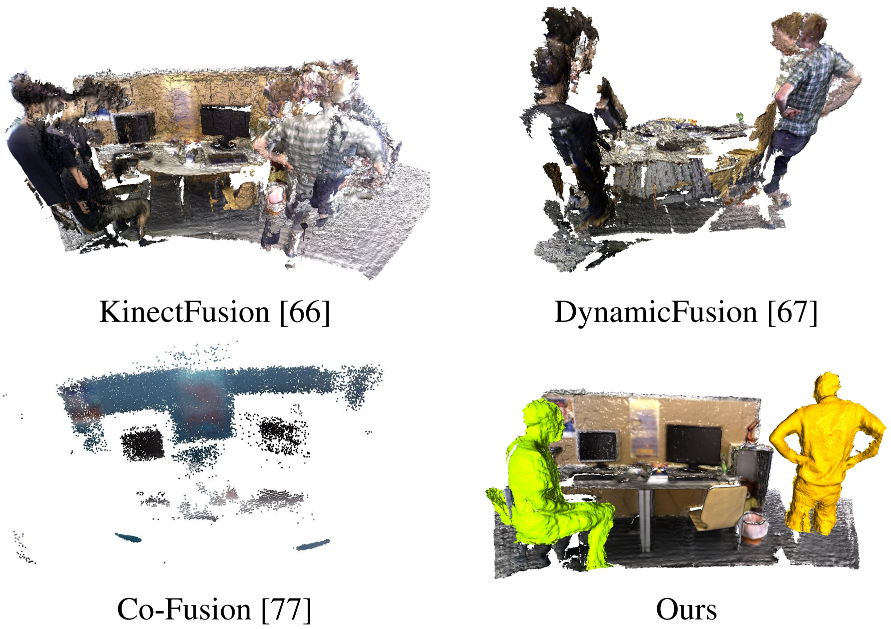


Figure 5.4 Reconstructions of non-rigid scenes with SplitFusion; top: meshes, bottom: normal maps; both the people and the camera are moving. From left to right: frame 1, 20 and 100 of the sequence fr3/walking_xyz from TUM RGB-D [87] dataset.


KinectFusion [66]

DynamicFusion [67]

Co-Fusion [77]

Ours

Figure 5.5 Reconstruction results on fr3-walking-xyz sequence from the TUM-RGBD [87] dataset. KinectFusion does not work when the scene is non-rigid. DynamicFusion fails to deal with the topology changes without explicit scene segmentation. Co-Fusion can not handle non-rigid objects.

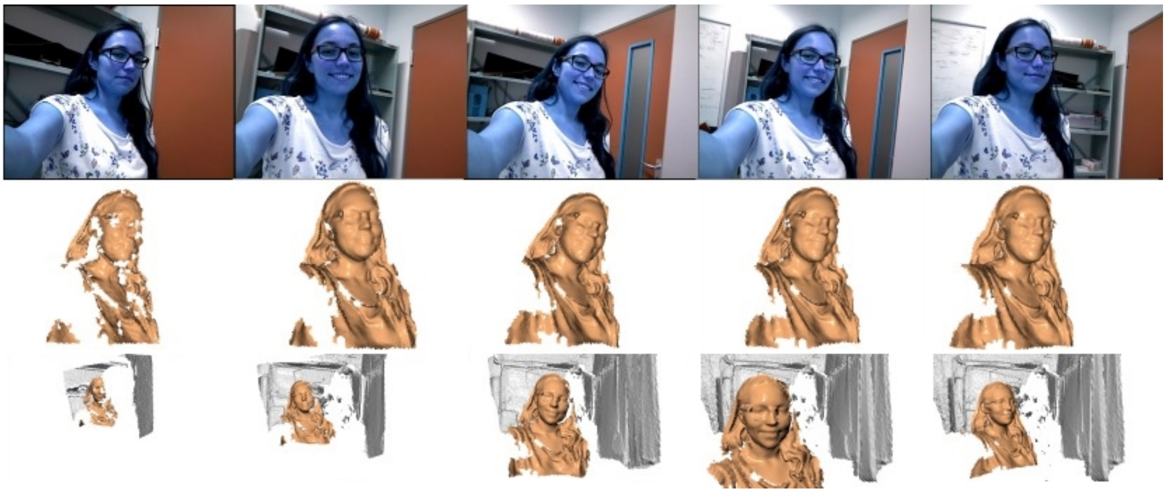


Figure 5.6 Our reconstruction results of the selfie sequence [80], from left to right, frame ID: 1, 39, 72, 88, 139, 168. Top row: the Color image; Middle row: reconstruction of the non-rigid face; Bottom row: reconstruction of the whole scene, including both rigid and non-rigid components. For the whole sequence results, we refer to the supplementary mp4 video.

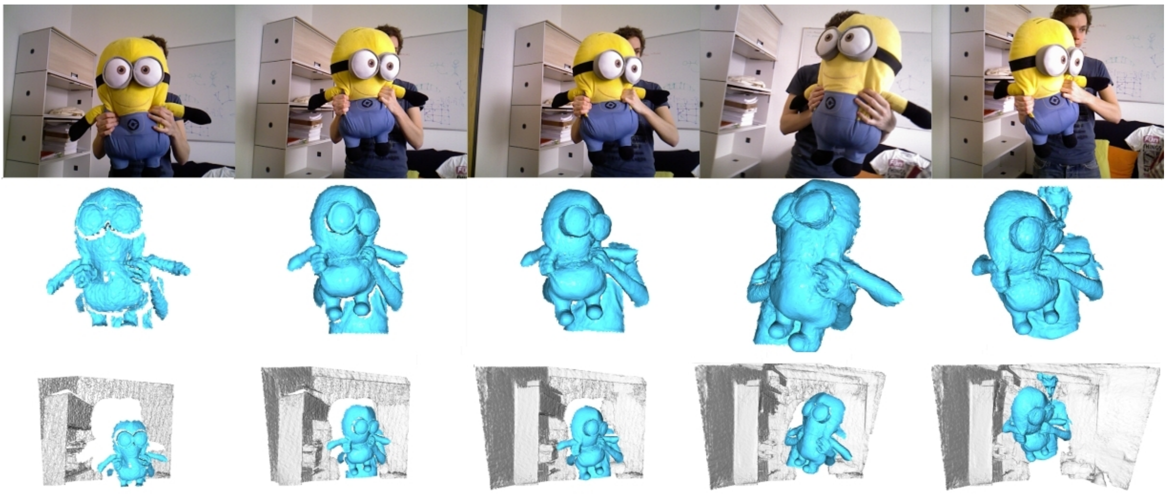


Figure 5.7 Our reconstruction results of the minions sequence [41], from left to right, frame ID: 3, 50, 100, 200, 426. Top row: the Color image; Middle row: reconstruction of the non-rigid objects; Bottom row: reconstruction of the whole scene, including both rigid and non-rigid components. For the whole sequence results, we refer to the supplementary mp4 video.

the room. However, the arm and shoulder can not be reconstructed because the commodity RGB-D camera can only capture depth that is further than 0.5 meters.

Minion This sequence is published by VolumeDeform [41]. This is also a very hard sequence. In this sequence, a person carries a “Minion" in front of a camera. A large portion of the frame is occupied by the non-rigid objects. The self-occlusion between the minions and human also make non-rigid tracking very hard. For the high entanglement, we treat the person and minion as a single object. The result of this sequence is shown in Fig. 5.7.

As shown in Fig. 5.4, Fig. 5.6, and Fig. 5.7 (from left to right), just like the results demonstrated in [67] [41], noisy and incomplete reconstruction can be progressively denoised and completed over time as more depth maps are fused into the model.

5.3.2 Camera pose tracking w.r.t the rigid background

To evaluate the rigid mapping results, we compared the proposed method to three state-of-the-art dynamic environment reconstruction methods: Co-Fusion(CF), StaticFusion(SF) and FlowFusion(FF). Tab. 5.1 shows the translational Absolute Trajectory Errors (ATE) on TUM[87] and HRPSlam[109] RGB-D datasets. The first two fr1 sequences are static environments, all of these methods got similar results. The lower four rows are dynamic sequences. As our method applied advanced dynamic object detection and removal technique, VO errors are much smaller than CF and competitive to SF and FF. The proposed method results in the smallest 4.8 cm ATE in fr3/walking_xyzsequence, the reconstructed maps are shown in Fig. 5.5.

Sequence	CF	SF	FF	Ours
fr1/xyz	0.014	0.017	0.020	0.015
fr1/desk2	0.17	0.051	0.034	0.040
fr3/walk_xyz	0.71	0.21	0.12	0.048
fr3/walk_static	0.58	0.031	0.23	0.21
HRPSlam2.1	0.91	0.25	0.23	0.22
HRPSlam2.4	0.63	0.44	0.49	0.45

Table 5.1 Absolute Trajectory Error (ATE) RMSE (m)

CF works well in known environments. There are moving objects in this sequence from the beginning, which result in CF’s VO failure. KF and DF cannot distinguish rigid and non-rigid objects, thus they cannot decouple the camera motion and failed in non-rigid tracking. The proposed SplitFusion split rigid and non-rigid objects. The rigid fusion pipeline provides an accurate static environment mapping and camera tracking(see the reconstructed desk and ground in the figure). Furthermore, the non-rigid fusion pipeline successfully tracks and reconstructs multiple non-rigid objects(see the yellow and green humans in the figure).

5.4 Summary

In this chapter, we proposed SplitFusion, which simultaneously tracks and reconstructs the rigid backgrounds and the deformable moving objects. Experimental results show that we can reconstruct accurate static environment maps and multiple non-rigid objects even in the challenging dynamic scenes.

5.5 Limitation

The following challenges are yet to be addressed:

1) the system requires excessive computation and memory on complex scenes with multiple objects, which could be alleviate by using light surfel as 3D representation,

2) both rigid and non-rigid tracking are not reliable on large motions, a promising direction is to incorporate the robust trackers [112][9][53][8].

3) in complex situations such as objects overlap/interact with each other, instance segmentation is not reliable anymore; to overcome such problem, a potential solution is to leverage the non-rigid geometry tracking information to improve instance segmentation.

4) The segmentation part of the system is based on YOLACT, which is a supervised object segmentation method that requires a large amount of data for training. Moreover, YOLOACT does not inform us which object is dynamic which is not. The solution is to leverage a self-supervised method to segment dynamic objects from static ones by reasoning the motion of the scene. Dynamic objects are outliers of rigid tracking. Keller et al. [46] has shown that when performing ICP for rigid tracking, failure of data association to find model correspondences for input points is a strong indication that these points are depth samples belonging to dynamic objects.

5.6 Supplementary material

5.6.1 Volumetric integration

Fig. 1.6 shows an example of global TSDF with truncation of 3. The global fusion of all depth maps in the volume is formed by the weighted average of all individual TSDFs computed for each depth map, which can be seen as de-noising the global TSDF from multiple noisy TSDF measurements. Under an L2 norm the denoised (fused) surface results as the zero-crossings of the point-wise TSDF F minimizing:

$$\min_{TSDF(p)} \sum_k ||W_k^{obs}(p) TSDF_k^{obs}(p) - TSDF(p)||_2 \tag{5.8}$$

where, $TSDF_k^{obs}(p)$ represent the TSDF of a voxel p given the depth observation from frame k, and $W_k^{obs}(p)$ stands for the weights of this frame.

Given that the focus of the system is on real-time sensor tracking and surface reconstruction we must maintain interactive frame-rates. (For a 640x480 depth stream at 30fps the equivalent of over 9 million new point measurements are made per second). Storing a weight $W_k^{obs}(p)$ with each value allows an important aspect of the global minimum of the convex L2 de-noising metric to be exploited for real-time fusion; that the solution can be obtained incrementally as more data terms are added using a simple weighted running average, defined point-wise:

$$TSDF_k(p) = \frac{W_{k-1(p)} TSDF_{k-1}(p) + W_k^{obs}(p) TSDF_k^{obs}(p)}{W_{k-1(p)} + W_k^{obs}(p)} \tag{5.9}$$

$$W_{k(p)} = W_{k-1(p)} + W_k^{obs}(p) \tag{5.10}$$

No update on the global TSDF is performed for values resulting from unmeasurable regions specified in Equation 9. While $W_k^{obs}(p)$ provides weighting of the TSDF proportional to the uncertainty of surface measurement, in practice simply letting $W_k^{obs}(p) = 1$, resulting in a simple average, provides good results. Moreover, by truncating the updated weight over some value

$$W_{k(p)} = \min(W_{k-1(p)} + W_k^{obs}(p), W_\eta) \tag{5.11}$$

a moving average surface reconstruction can be obtained enabling reconstruction in scenes with dynamic object motion.

Chapter 6

Structure and Motion Completion for Non-rigidly Deforming Scene

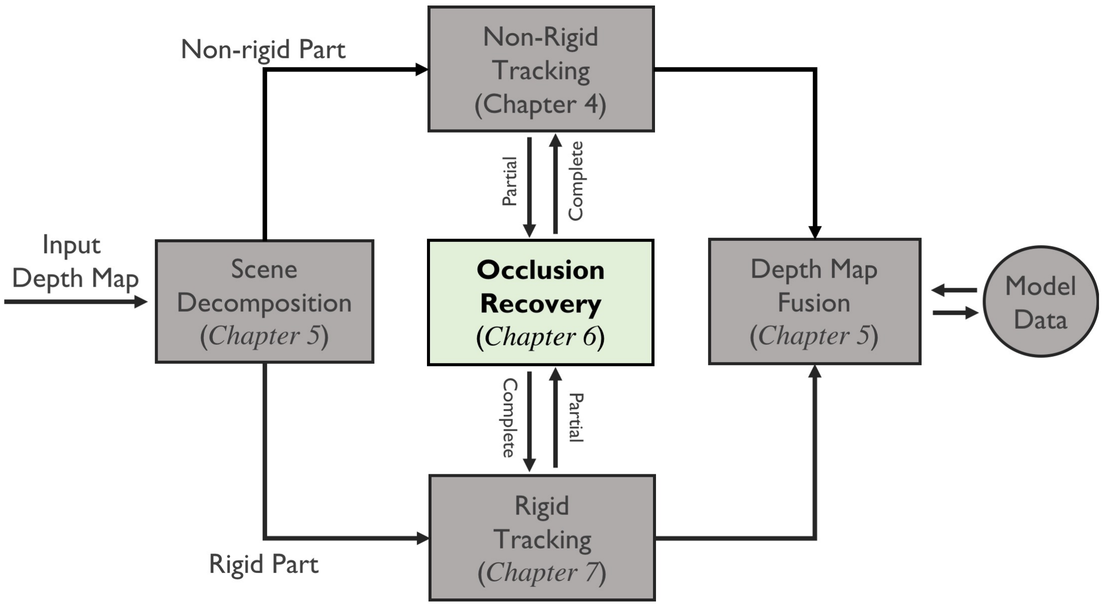

Non-rigid Part
Non-Rigid Tracking (Chapter 4)
Partial
Complete
Input Depth Map
Scene Decomposition (Chapter 5)
Occlusion Recovery (Chapter 6)
Depth Map Fusion (Chapter 5)
Model Data
Complete
Partial
Rigid Part
Rigid Tracking (Chapter 7)

Figure 6.1 The focus of this chapter and its position in the entire system.

As shown in Figure. 6.1, this chapter focuses on how to handle the occlusion of the scene during non-rigid or rigid tracking. Tracking non-rigidly deforming scenes using range sensors has numerous applications in computer vision, virtual/augmented reality, and robotics, etc. However, due to occlusions and physical limitations of range sensors, existing methods can only handle the visible surface, leading to discontinuity and incompleteness in the geometry and motion field. In this chapter, we introduce a novel data-driven approach that estimates the non-rigid motion for the occluded geometry.

6.1 Introduction

Motion represents how structure evolves in the time axis. Understanding the motion of non-rigidly deforming scenes using a single range sensor is an important task for many computer vision and robotics applications. One fundamental limitation is that, due to occlusions and the physical limitations of range sensors, a single depth camera can obtain only partial and incomplete observations of a given scene, see Fig. 6.2. As a result, existing non-rigid motion tracking methods are restricted to the observable part of the scene. However, the ability to infer complete motion from partial observation is indispensable for many high-level tasks. For instance, as a domestic robot, to safely serve in a room, it needs to understand the 3D structure and complete motion field for all the dynamic objects in the room (i.e. human and pets), even that the objects are always partially occluded.

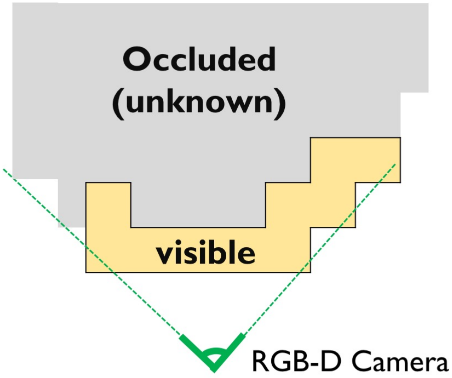


Figure 6.2 The illustration of occlusion in an RGB-D camera. The ray from the camera terminates at the visible surface of a given scene, beyond that is occluded and unknown.

Motion estimation of the hidden surface is a widely encountered problem in non-rigid 4D reconstruction using depth sensors. Existing works such as DynamicFusion [67] and Volumedeform [41] try to tackle it by propagating deformation from the visible surface to the invisible space through a latent deformation graph. The invisible deformation is determined by optimizing the hard-coded deformation priors such as As-Rigid-As-Possible [85] or Embedded Deformation [88], which enforces the graph vertices locally move in an approximately rigid manner. Such deformation priors have several limitations: 1) They require heavy parameter tuning effort; 2) do not always reflect nature deformation; and 3) work only on a connected surface. Occlusion causes surface disconnections and therefore poses problems for non-rigid tracking, see Fig. 6.3.

One promising direction towards solving this problem is to use machine learning for shape completion. Very recently, deep learning approaches for 3D shape or scene completion

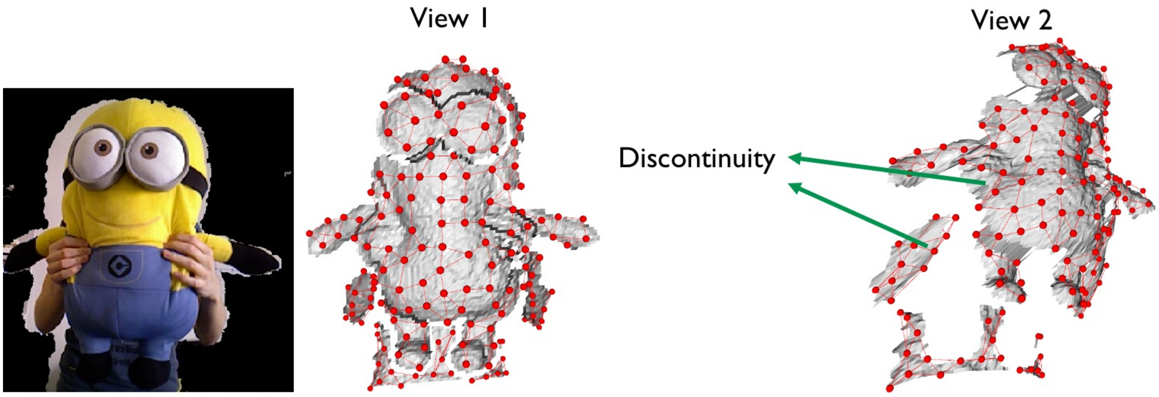


Figure 6.3 Occlusion causes surface disconnections and therefore poses a problem for non-rigid tracking. Left: the input color image; Middle: the surface deformation graph sampled from the input surface (nodes are connecting to nearby nodes within a geodesic distance threshold, c.f. Chapter 14 for deformation graph construction); Right: the surface visualized from another viewpoint (due to occlusion, the hand and arm of the person are not covered by the same graph, in this case, the motion of the two graphs have to be tracked separately).

and other generative tasks involving a single depth image or room-scale scans have shown promising results [20, 84, 21, 22]. However, these works focused on processing static scenes.

In this chapter, we make the first effort to combine geometry completion with non-rigid motion tracking. In nature, the structure and motion of non-rigidly deforming objects are highly entangled data modalities. It has been long recognized that natural deformations lie on a spatial-temporal submanifold [10, 23, 50]. In addition, we argue that structure completion and motion estimation are two mutually complementary tasks. On one hand, natural objects are usually articulated, e.g. animals and humans. The surface deformation could be a blending result of several articulations. The other way around, motion can be considered as the structure's evolution in the time axis, the similarity in motion patterns is a strong indicator for structural connectivity. To bridge the best of the two, we propose a novel framework to jointly recover the missing geometry and predict the invisible motion, see Fig. 6.4. To train our network, we generate training data from animated CG characters (c.f. Section. 6.3.2).

6.2 Related work to geometry completion

Completing 3D scans has been well-studied in geometry processing. Traditional methods, such as Poisson Surface Reconstruction [45], locally optimize for a surface to fit observed points and work well for small missing regions. Zheng et al [111] predict the unobserved

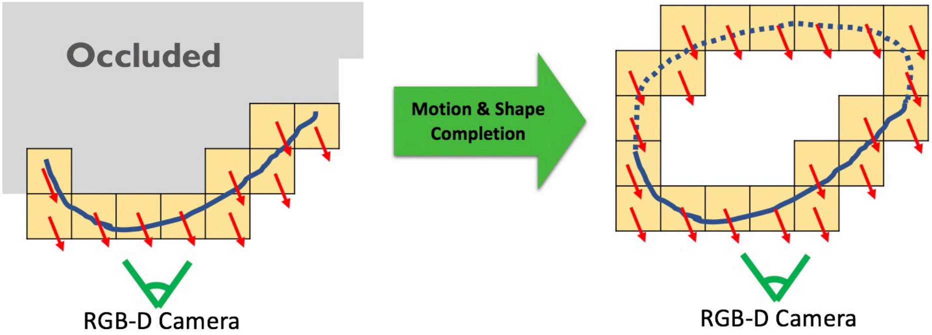


Figure 6.4 We propose to jointly recover the missing geometry and predict the invisible motion from partial observation. In our method, we represent both the shape and motion field using 3D volume grid.

voxels by reasoning physics. Halimi et al [38] complete partial human scans by deforming human templates. 3D CNNs have shown promising results for geometry completion for depth scans [84, 20–22]. Among these works, SSCNet [84], operating on a depth image of a scene; ScanComplete [21] and SGNN [22] demonstrated scene completion on room- and building floor-scale scans. An alternative approach for shape completion could through learning implicit scene representations [63, 68, 73, 70, 54, 15, 44, 82]. While existing works mainly focus on completing static scenes, we investigate how to do completion in the dynamic 4D domain.

6.3 Method

6.3.1 Network Configuration

Given a single-view depth map observation of a 3D scene, and the scene flow that is computed between the current frame and the next frame, the goal of our network is to recover the missing geometry and motion filed.

Input. We use 3D volumetric representation for both structure and motion. The input depth map is represented as a projective truncated signed distance field (TSDF), as a sparse set of voxel locations within truncation and their corresponding distance values. The input scene flow is the 3D translational vectors of the 3D points that are reprojected from the depth map's pixels. In this work, we use Flownet3D [55] to predict the scene flow for the visible surface. Because a 3D point does not necessarily lie on a regular 3D grid position, we voxelize the sparse scene flow field to a 3D volumetric motion field. We bind the flow motions to the voxels using Eqn. 6.2. The projective TSDF and the voxelized scene flow are concatenated together as network input.

Network Architecture. To obtain an output high-resolution structure and motion field, we build the networks upon the sparse convolutions [34, 35, 16], which is computationally efficient in processing 3D volume data by operating only on sparse surface geometry. Fig. 6.5 shows an overview of our network architecture. The network consists of a shared 4D encoder and two decoders to estimate geometry and motion in parallel. The input sparse tensor is first fed to the *4D Encoder*, which encodes the data using a series of sparse convolutions, each set reduces the spatial dimensions by a factor of two. Similar to [22], the structure decoder is designed in a coarse-to-fine architecture, with 4 hierachical levels. In the hierarchical level k, the structure decoder predicts the voxels' occupancy O_k and TSDF value S_k. We filter voxels with $sigmoid(O_k(v)) > 0.5$ as the input geometry for the next hierarchical level. In each hierarchical level, the predicted geometry is also fed to the parallel motion decoder, i.e. to inform where the motion should be estimated. We use skip connections between the 4D encoder and the 2 decoders to connect feature maps of same spatial resolution. Since the structure decoder usually generate a larger set of sparse locations than the input, we use zero feature vector for the locations that do not exist in the input volume.

Structure Loss. The structure head's final output is a sparse TSDF from which a mesh can be extracted by Marching Cubes. Following [22], we apply $l1$ loss on the log-transformed TSDF values. The log-transformation encourage more accurate prediction near the surface geometry. We additionally employ proxy losses at each hierarchy level for outputs O_k and S_k, using binary cross entropy with target occupancies and $l1$ with target TSDF values, respectively.

Motion Loss. For each sparse location, the motion head predicts its motion vector in $\mathbb{R}^3$. We formulate the loss for the completed motion field on the final predicted sparse locations, using an $l2$ loss with the target motion vectors at those locations. In addition, we apply the cosine similarity loss on the normalized motion vector to encourage the directions of the motion vectors to be consistent with the ground truth.

Training. We use a newly constructed dataset (c.f. Section. 6.3.2) to train our network. As shown in Fig. 6.5, at training time, we consider cropped views of scans for efficiency, using random crops of size $[96 \times 96 \times 128]$ voxels for the finest level. We crop the volumes at 1 meter intervals out of each of the train object and discard empty volume. The resolution drops by a factor of 2, resulting resolution of $[48 \times 48 \times 64]$, $[24 \times 24 \times 32]$, and $[12 \times 12 \times 16]$ for each hierarchical level. The fully-convolutional nature of our approach enables

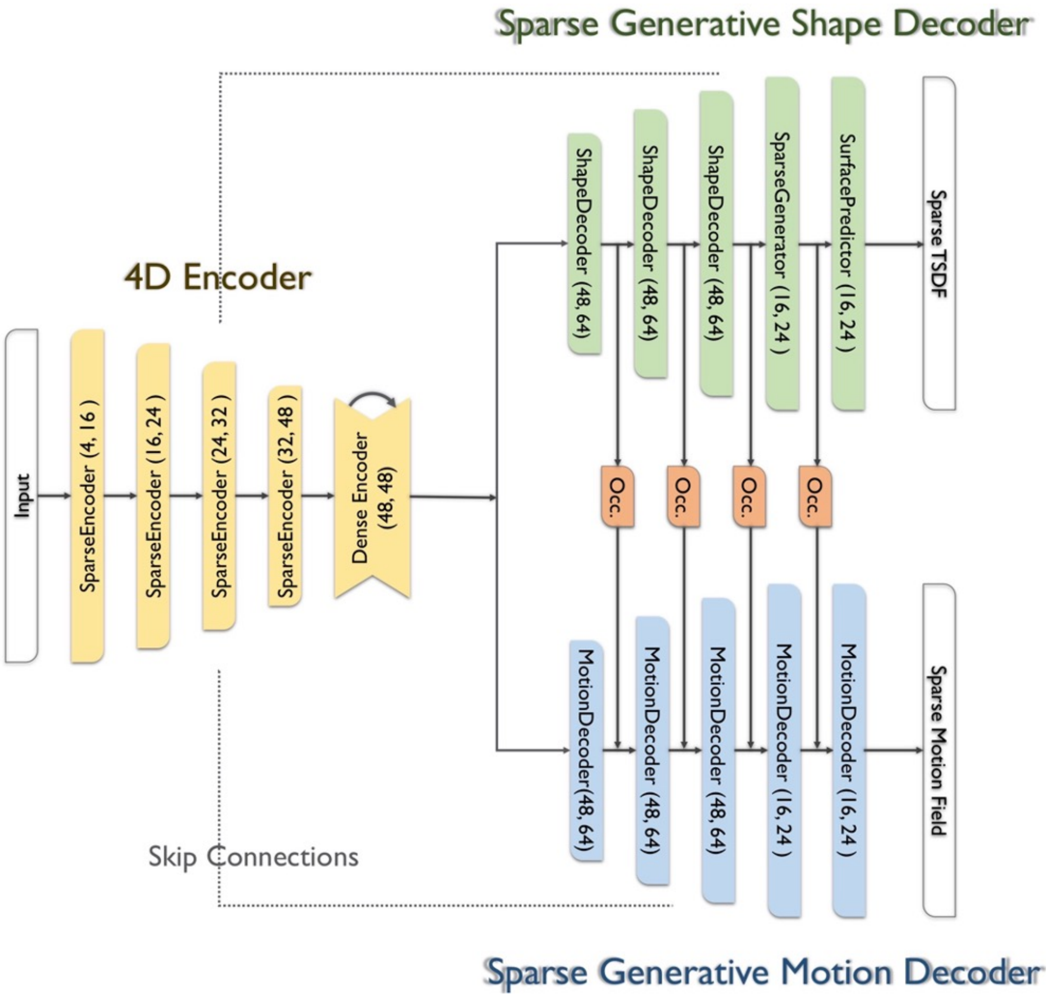


Figure 6.5 The network architecture of our method. The input of our network is the concatenation of a TSDF volume and volumetric motion field. The network predicts the complete TSDF and motion field using the shape and motion decoder in parallel. Our method is trained on the cropped volume of spatial dimension $96 \times 96 \times 128$. The fully-convolutional nature of our approach enables testing on whole objects of arbitrary sizes at testing time.

testing on whole object of arbitrary sizes at testing time. To learn viewpoint invariant motion representation, during training, we apply random rigid rotation transformation on the 3D motion vectors as data-augmentation. The randomness is drawn from the Haar distribution [86] which yield uniform distribution on SO3. We train our network, using the Adam optimizer with a learning rate of 0.001 and batch size of 8. We use level 2000 iterations for progressive introduction of each higher resolution output, and train our model for 40 hours until convergence.

6.3.2 Mixamo Animation dataset

Training our network requires a sufficient amount of non-rigidly deforming target sequences with ground truth 4D annotation, ($i.e.$motion and shape) at the voxel level. However, such a dataset does not exist. To overcome the dataset issue, we construct a synthetic dataset, which consists of a large number of animated humanoids characters with skin mesh, texture, and skeleton. Generally, these characters are animated by using “rigging" and “skinning" to blend the skeletal movement to the surface skin mesh. We borrow the humanoid characters from Adobe Mixamo[1], the motion of which was collected using the motion capture system. The dataset contains 100 different humanoids characters and 200 animated sequences. From Mixamo animated characters, we generate ground truth for per-frame TSDF, per-frame Volumetric Motion Field, scene flow, and RGB-D image.

Crop volume for training. At training time, as shown in Fig. 6.6, we consider cropped views of scans for efficiency, using random crops of size $[96 \times 96 \times 128]$ voxels for the finest level. We crop the volumes at 1 meter intervals out of each of the train object and discard empty volume. As shown in Fig. 6.7, the resolution drops by a factor of 2, resulting resolution of $[48 \times 48 \times 64]$, $[24 \times 24 \times 32]$, and $[12 \times 12 \times 16]$ for each hierarchical level. The fully-convolutional nature of our approach enables testing on whole object of arbitrary sizes at testing time.

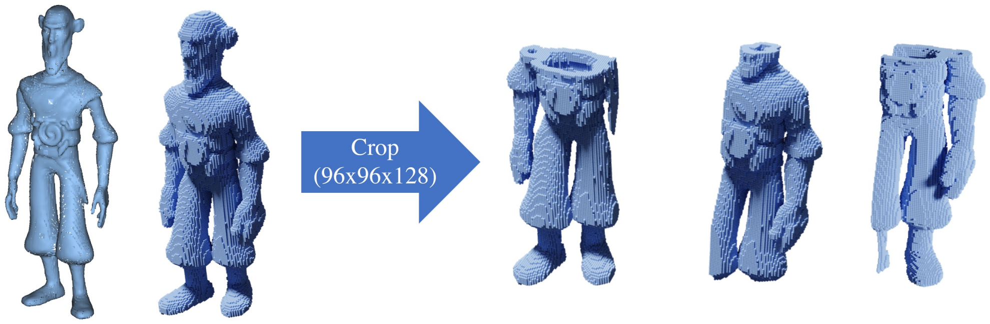


Figure 6.6 At training time,we consider cropped views of scans for efficiency, using random crops of size $[96 \times 96 \times 128]$ voxels for the finest level. We crop the volumes at 1 meter intervals out of each of the train object and discard empty volume. Left: original mesh and volume. Right: cropped volumes.

[1] https://mixamo.com

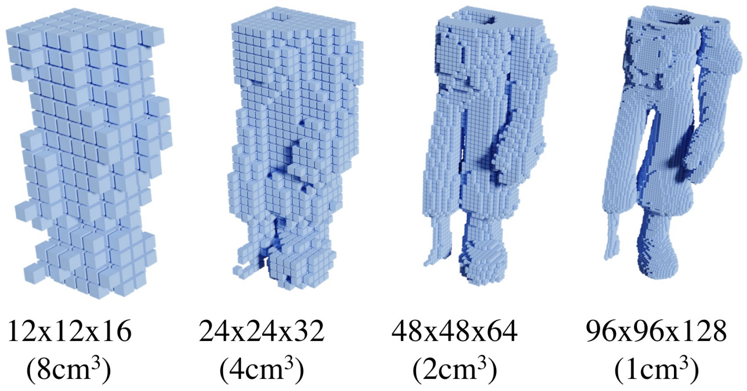


Figure 6.7 Four hierarchy levels of the cropped volumes with voxel sizes of $1.0cm^3$, $2.0cm^3$, $4.0cm^3$, and $8cm^3$.

6.4 Experimental results

Scene Flow Field ⇔ Volumetric Motion Field. Here we show how we do the conversion between the scene flow field (SFF) and the volumetric motion field (VMF). Given a point cloud $\{p_i | i = 1, ..., N\}$, where $p_i \in \mathbb{R}^3$ are XYZ coordinates of individual point, the SFF is defined as $\{\mathscr{SFF}_i | i = 1, ..., N\}$, where $\mathscr{SFF}_i \in \mathbb{R}^3$ are the 3D translational motion vectors of the points. Similarly, given a set 3D voxel positions $\{v_j | j = 1, ..., M\}$, the VMF is defined as $\{\mathscr{VMF}_i | i = 1, ..., M\}$, where $\mathscr{VMF}_i \in \mathbb{R}^3$ are the 3D translational motion vectors of the voxels. To convert from SFF to VMF, we use the inverse-distance weighted interpolation as defined in [75]:

$$\mathscr{VMF}_j = \sum_{p_i \in knn(v_j)} \frac{\mathscr{SFF}_i \cdot dist(p_i, v_j)^{-1}}{\sum_{p_i \in knn(v_j)} dist(p_i, v_j)^{-1}} \tag{6.1}$$

where $knn()$ is the fucntion to find K-Nearest-Neighbors, here we set neighbor number $K = 3$, and $dist(,)$ computes the euclidean distance between two positions. To convert from VMF to SFF, we do tri-linear interpolation:

$$\mathscr{SFF}_j = \sum_{v_j \in knn(p_i)} \mathscr{VMF}_j \cdot w(p_i, v_j) \tag{6.2}$$

where $w(,)$ computes the linear-interpolation weights, and $K = 8$ represent the neigboring 8 corner voxels of the cube that the point lies in. Note that throughout the experiments, we convert all VMF to SFF before doing motion evaluation.

Motion Evaluation Metric. Following [55], we use 3D end point error (EPE) and motion accuracy (ACC) as our motion evaluation metrics. The 3D EPE measures the average euclidean distance between the estimated motion vector to the ground truth motion vector. The ACC score measures the portion of estimated motion vectors that are below a specified end point error, among all the points. We report two ACC metrics with different thresholds.

Structure Evaluation Metric. To measure structure completion quality, we follow [21] and use an l_1 error metric between predicted and target TSDFs, where unobserved regions in the target are masked out. Note that unsigned distances are used in the error computation to avoid sign ambiguities. We measure the l_1 distance in voxel units of the entire volume (*entire volume*), and the unobserved region of the volume (*unobserved space*). We use a global truncation of 3.

6.4.1 Structure completion results

This section we show qualitative structure completion results of our approach. Fig. 6.8 shows the structure completion results for RGB-D sequences from [41, 53]. Fig. 6.9 shows the structure completion results for RGB-D sequences from MPI-Sintel [12]. Since our method is based on fully convolutional sparse CNN, it can test on the large Sintel scene (the max depth is 20 meters in this Sintel frame). Fig. 6.10 shows the testing results for synthetic characters in Mixamo dataset.

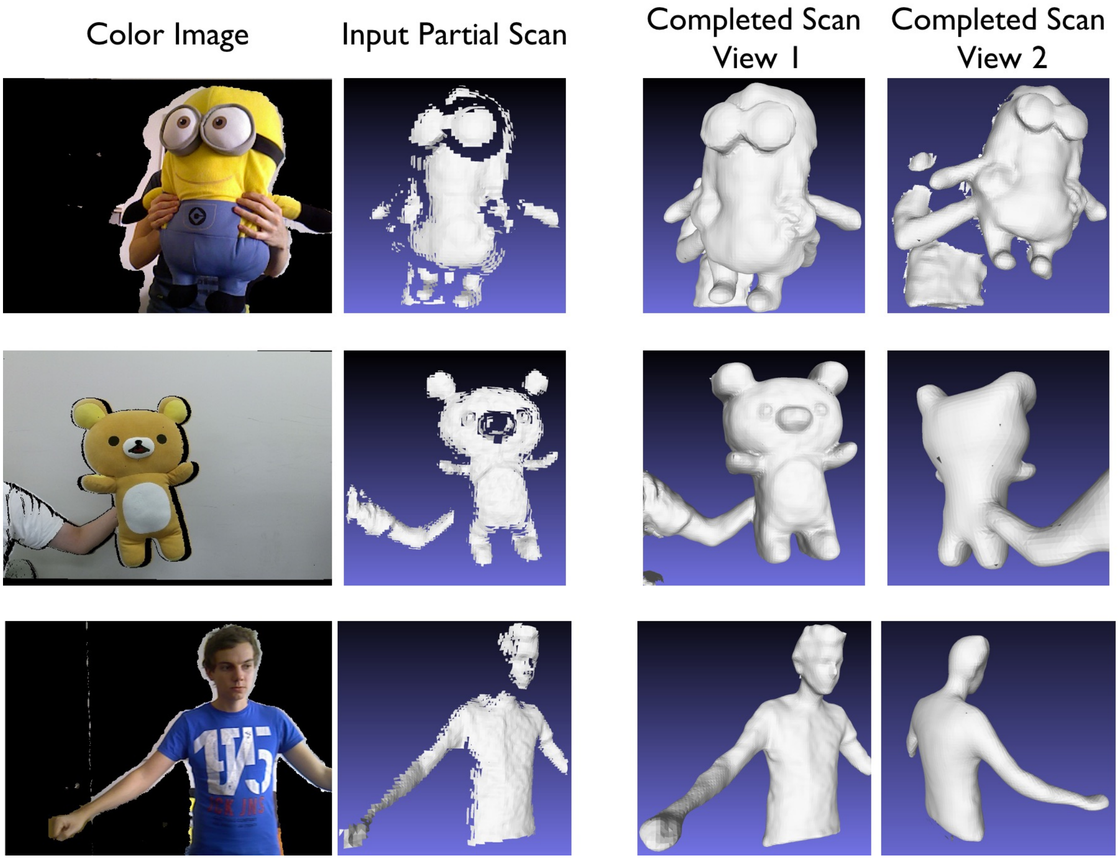


Figure 6.8 Structure completion results on real world RGB-D image.

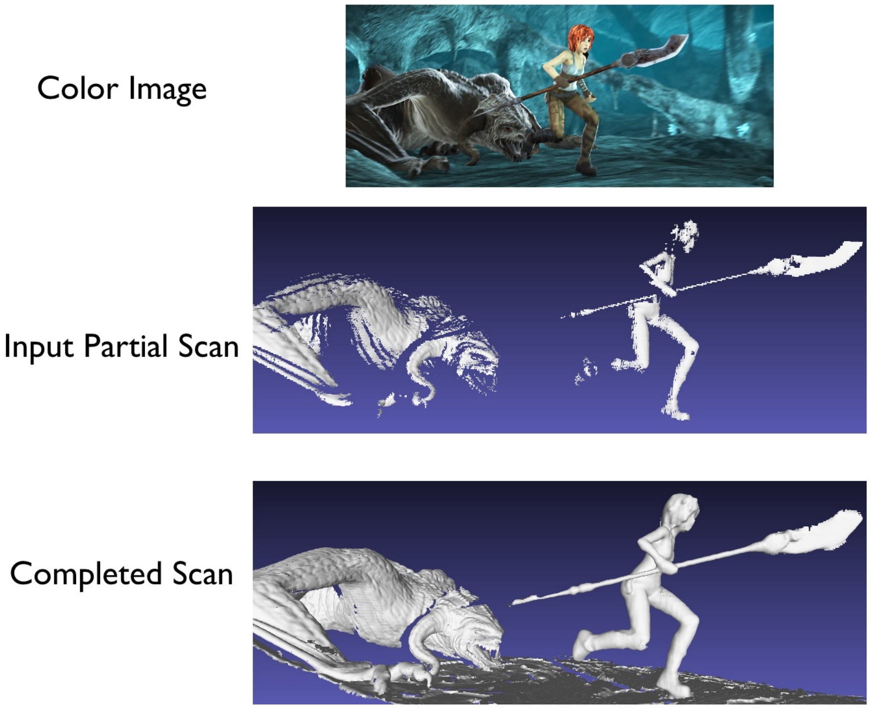


Figure 6.9 Structure completion results on MPI Sintel [12] dataset. The fully convolutional nature of our method allows testing on very large scene (The max depth is 20 meters in this Sintel frame).

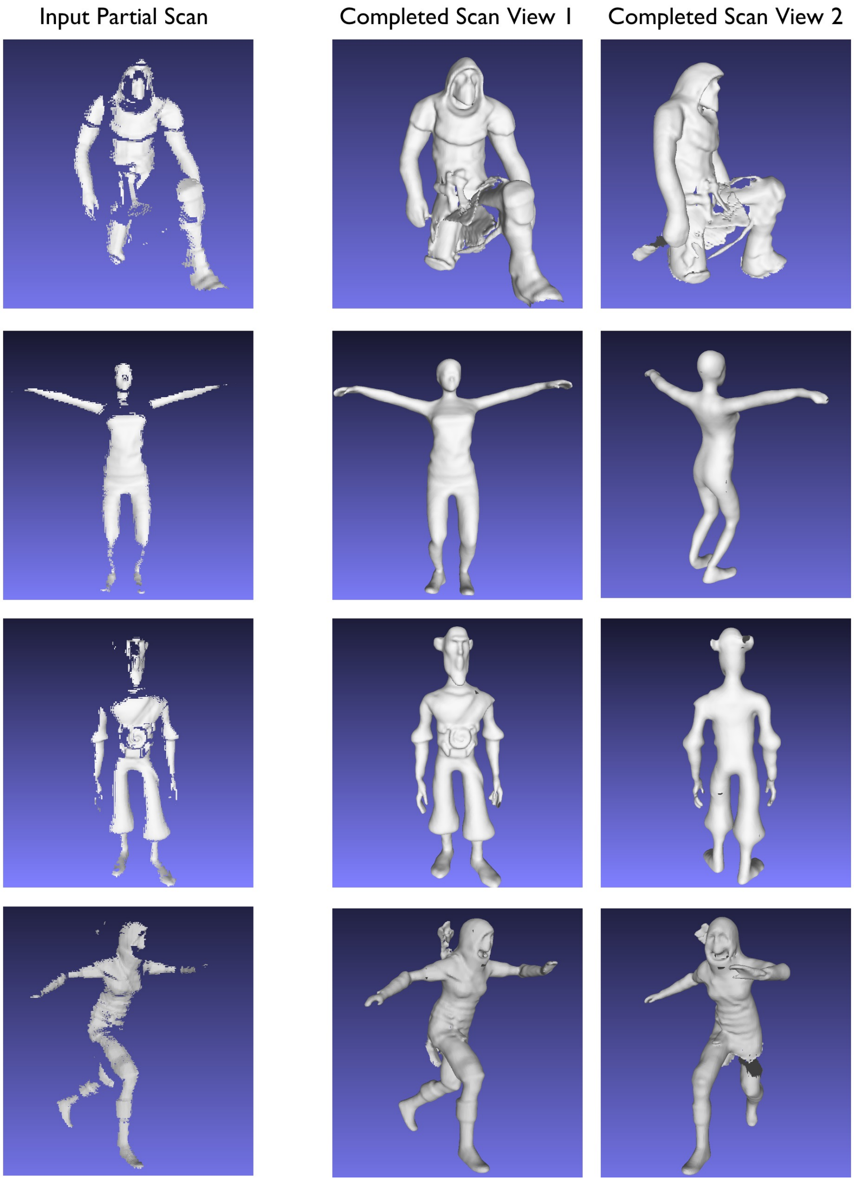


Figure 6.10 Structure completion results on synthetic dataset. The examples are not included in the training dataset. This result also reveal two major limitations of the proposed completion method. 1) As shown in the 3rd row: the completion model can retain geometry details from the visible surface but can not generate geometry details for the occluded surface. 2) Our model can not reconstruct the shape for heavily occluded areas, e.g. in the 4th row, the left leg of this person (a promising direction is to leverage skeleton information for articulated objects).

6.4.2 Ablation study of Network Architecture

Method	SGNN [22]	Ours (*no motion*)	Ours (*joint*)
l_1-Err	0.529	0.531	**0.503**

Table 6.1 Surface prediction error on testing set of synthetic. SGNN [22] is retrained on this dataset. We measure the l_1 error against the ground truth distance field in voxel space for the entire space with 2cm voxels and global truncation of 3.

This experiment examines how the two tasks: geometry completion and motion tracking influence each other. To this end, we train the following 3 model: "*Joint*", "*no. completion*", and "*no motion*". As defined in Fig. 6.5, "*Joint*" is the full model that perform both tasks. "*no completion*" remove the structure decoder and only predict the motion for the observed surface. "*no motion*" removes the motion decoder and only do geometry completion. We pre-train the Flownet3D [55] on the scene flow data. FlowNet3D predicts the SFF given a pair of point clouds with a subsampled size of 2048. We convert the sparse SFF to VMF using Eqn. 6.2 as network input. The voxel position in the VMF is consistent with the input projective TSDF.

Does geometry completion help non-rigid tracking? Table. 6.2 reports the motion prediction results for the visible surface. Though only evaluating on the visible surface, the model trained with the added supervision of the geometry completion task show improvement over the model trained only on motion prediction. This demonstrates that understanding complete object geometry is beneficial for non-rigid tracking.

Does motion prediction help geometry completion? Table. 6.1 reports the geometry completion results in the synthetic synthetic dataset. The "*joint*" model show improvement over the model that is train for geometry completion only. This result validates the idea that in dynamic scene it is beneficial to understand the motion in order to achieve better geometry completion.

6.4.3 Motion prediction for the invisible surface

This section evaluate motion estimation of the unobserved surface. We conduct the following experiment: the baselines are given a complete mesh, a subset of mesh vertices that are observed from a given camera viewpoint, and the scene flow for the observed vertices. The goal is to estimate the motion of the unobserved vertices. (Note that this is a widely encountered situation in non-rigid reconstruction. For instance, DynamicFusion [67] always

Method	Training dataset	Input	Mixamo Animation		
			EPE↓	ACC(5)↑	ACC (10)↑
FlowNet3D	FT3D [62]	points	7.36	69.43%	80.04%
FlowNet3D	Deforming Things4D	points	3.74	82.02%	**91.63**%
Ours (*no completion*)		VMF	3.82	79.02%	90.55%
Ours (*join*)		VMF	**3.56**	**85.02**%	91.59%

Table 6.2 Scene Flow estimation results on the synthetic (FT3D) dataset. Note that all scores are reported only for the visible surface. Metrics are End-point-error (EPE) in centimeter, and Acc ($<5cm$ or 5%, $10cm$ or 10%) for motion. FT3D stands for Flyingthings3D [62] dataset.

needs to infer the deformation of the occluded surface when the target model turns around from the camera.)

We implement the following baselines.

•**Rigid Fitting**. This method assumes that the scene undergoes rigid motion. It finds a single rigid transform in $SE(3)$ for the entire mesh that best explains the surface motion.

•**As-Rigid-As-Possible (ARAP) Deformation**. ARAP [85] is widely used as the deformation prior in many non-rigid reconstruction method [67, 41, 114]. It assumes that locally a mesh vertex should be transformed with rigid transformation. Such rigid constraints are imposed upon nearby vertices that are connected by edges. ARAP deformation finds for each mesh vertex a local fan-rotation $R \in SO(3)$ and a global translation vector $t \in \mathbb{R}^3$ that best explains the anchor motion (*i.e.*scene flow on observed vertices) and local rigidity constraints. Refer to [85, 40] for more details.

•**Motion Complete (Ours)**. The mesh and scene flow are voxelized as the input to our network. The output of our motion complete network is a 3D volumetric motion field. The final motion of a mesh vertex is computed by trilinearly interpolating the motion vectors from 8-nearest voxels on the volume grid.

•**Motion Complete + Post Processing (PP) (Ours)**. We found that the raw output motion field is noisy. We employ optimization-based post-processing to alleviate the noise: the predicted motion filed on the mesh surface is jointly optimized with ARAP prior that enforce geodesic neighbors have similar motions.

Results. Table 1 reports the motion estimation results for the occluded surface. The testing sequence is the Samba Dance sequence from Mixamo dataset. Among the baselines, rigid fitting yields significantly larger error, which indicate that the sequences undergo large non-rigid motion. Our Motion Completion consistently achieves lower end-point-error than the ARAP. Motion Completion + PP further improves the numbers. Our take is that the network with large receptive field learns to capture the global deformation.

Methods	Mixamo testing sequence (Samba Dance)
Rigid Fitting	15.30
ARAP [85]	3.24
MotionCompletion (Ours)	2.32
MotionCompletion + PP (Ours)	**1.81**

Table 6.3 Quantitative evaluation for the motion estimation results of the unobserved surface. The Metric is 3D End-Point-Error (EPE) in centimeter. Note that our method is trained only on humanoid motions.

6.5 Summary

In this chapter, we introduce the first method that jointly recovers the high-resolution structure and motion field from partial observation. Ablation study shows that our method learns entangled 4D feature representation that benefits both structure and motion estimation. By deforming the occludede surface, our method yield more accurate deformation than classic non-rigid priors such as ARAP deformation. The experiments on DynamicFusion/VolumeDeform sequence, and MPI Sintel shows that our method generalizes well to unseen categories in real-world scans. We also show that synthetic dataset such as Mixamo can be a good option for training shape and motion completion network, the results also generalize to real-world scans.

6.6 Limitation

We believe this work is a significant step toward robust non-rigid reconstruction for generic scenes, several major challenges are yet to be addressed:

1) the method only uses the geometry information of a single frame, an interesting direction is to combine the geometry information from both two frames for more detailed shape completion.

2) the completed shape from 3D CNN is not necessarily water-tight, it easily generates un-filled holes in the heavily occluded region; see Figure 6.10. A promising direction is to use the implicit occupancy networks [63] for geometry completion, which treats the 3D

surface as a decision boundary between occupied and empty space, thus easily generate water-tight surfaces.

3) As shown in Fig. 6.10, the completion model can retain geometry details from the visible surface but can not generate geometry details for the occluded surface. The model can not reconstruct the shape for heavily occluded areas, e.g. in the forth row, the left leg of this person (a promising direction is to leverage skeleton information for articulated objects).

6.7 Supplementary material

In this document, we provide more details to the main chapter. In Sec. 6.7.1, we show how we get the scene flow for the visible surface. In Sec. 6.7.2, we provide more details about the the generated synthetic dataset.

6.7.1 Scene flow estimation for the visible surface

We use FlowNet3D [55] to estimate the scene flow between two consecutive RGB-D frames. As shown in Fig. 6.11, FlowNet3D directly operates on point clouds. It learns to predict the scene flow as translational motion vectors for each point of the first frame.

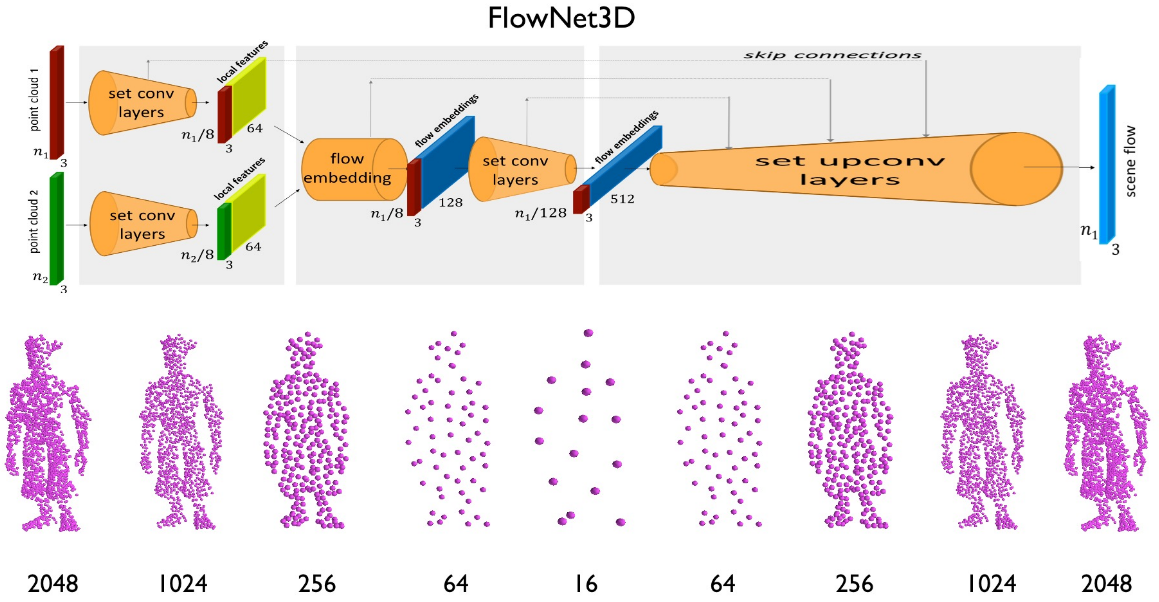
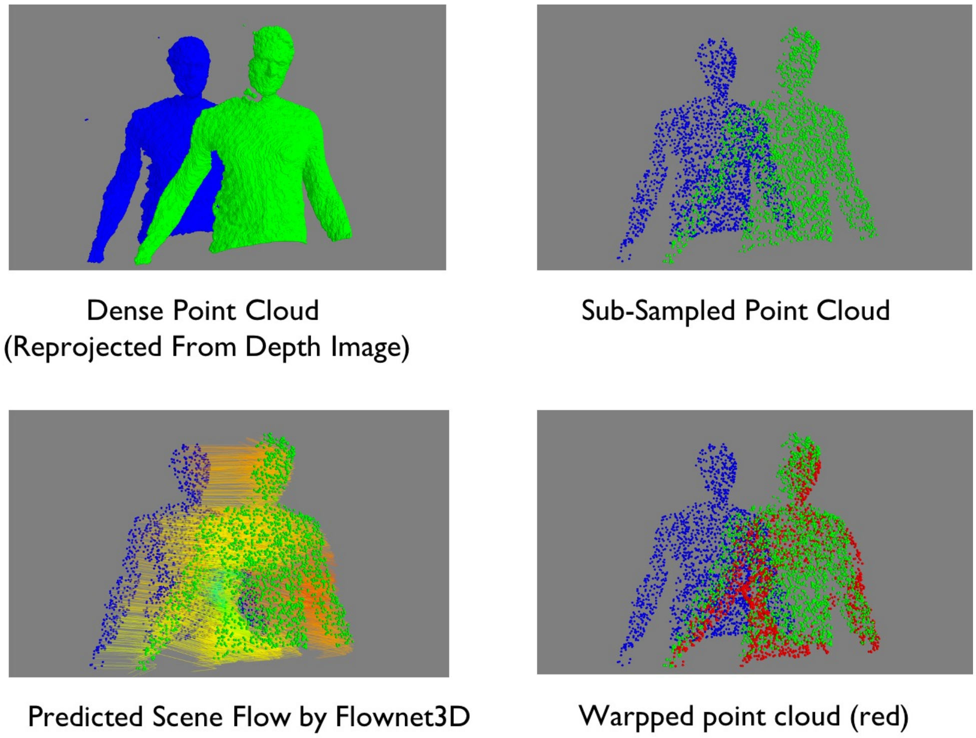


Figure 6.11 The network architecture of Flownet3D [55]. Given two frames of point clouds, the network learns to predict the scene flow as translational motion vectors for each point of the first frame. The input pointsets are subsampled to 2048 points. And through each encoder/decoder layers, it down-/up-sample the number of points by a factor of 2 or 4. The lower part shows an example of the point sampling through the layers. See the original paper [55] for more details.

The final FlowNet3D architecture is composed of four set conv layers, one flow embedding layer and four set upconv layers (corresponding to the four set conv layers) and a final linear flow regression layer that outputs the $\mathbb{R}^3$ predicted scene flow. For the set upconv layers we also have skip connections to concatenate set conv output features. Each learnable layer adopts multi-layer perceptrons for the function h with a few Linear-BatchNorm-ReLU

Layer type	*r*	Sample rate	MLP width
set conv	0.5	0.5×	[32, 32, 64]
set conv	1.0	0.25×	[64, 64, 128]
flow embedding	5.0	1×	[128, 128, 128]
set conv	2.0	0.25×	[128, 128, 256]
set conv	4.0	0.25×	[256, 256, 512]
set upconv	4.0	4×	[128, 128, 256]
set upconv	2.0	4×	[128, 128, 256]
set upconv	1.0	4×	[128, 128, 128]
set upconv	0.5	2×	[128, 128, 128]
linear	-	-	3*

Table 6.4 **FlowNet3D architecture specs.** Note that the last layer is linear thus has no ReLU and batch normalization.



Dense Point Cloud
(Reprojected From Depth Image)

Sub-Sampled Point Cloud

Predicted Scene Flow by Flownet3D

Warpped point cloud (red)

Figure 6.12 Example of running FlowNet3D on a pair of point clouds from real-world RGB-D images. Top Left: the dense point cloud that is reprojected from the depth images. Top Right: the sub-sampled point cloud input to FlowNet3D, each have 2048 points. Bottom Left: the estimated scene flow vectors which are visualized by the arrows. Bottom Right: warping the source point cloud to the target using the estimated scene flow.

layers parameterized by its linear layer width. The detailed layer parameters are as shown in Table 6.4. Fig. 6.12 show a example of running FlowNet3D on a pair of point clouds from real-world RGB-D images. Fig. 6.14 and Fig. 6.15 show the results of scene flow estimation on a real-world RGB-D video recorded from Azure Kinect camera.

6.7.2 Animation Dataset details

Fig. 6.16 shows the example of animated characters in the Adobe Mixamo project (https://mixamo.com). Fig. 6.13 shows the rendered data for the Mixamo "Aiming Gun" sequence, including color image, depth image, and inter-frame scene flow.

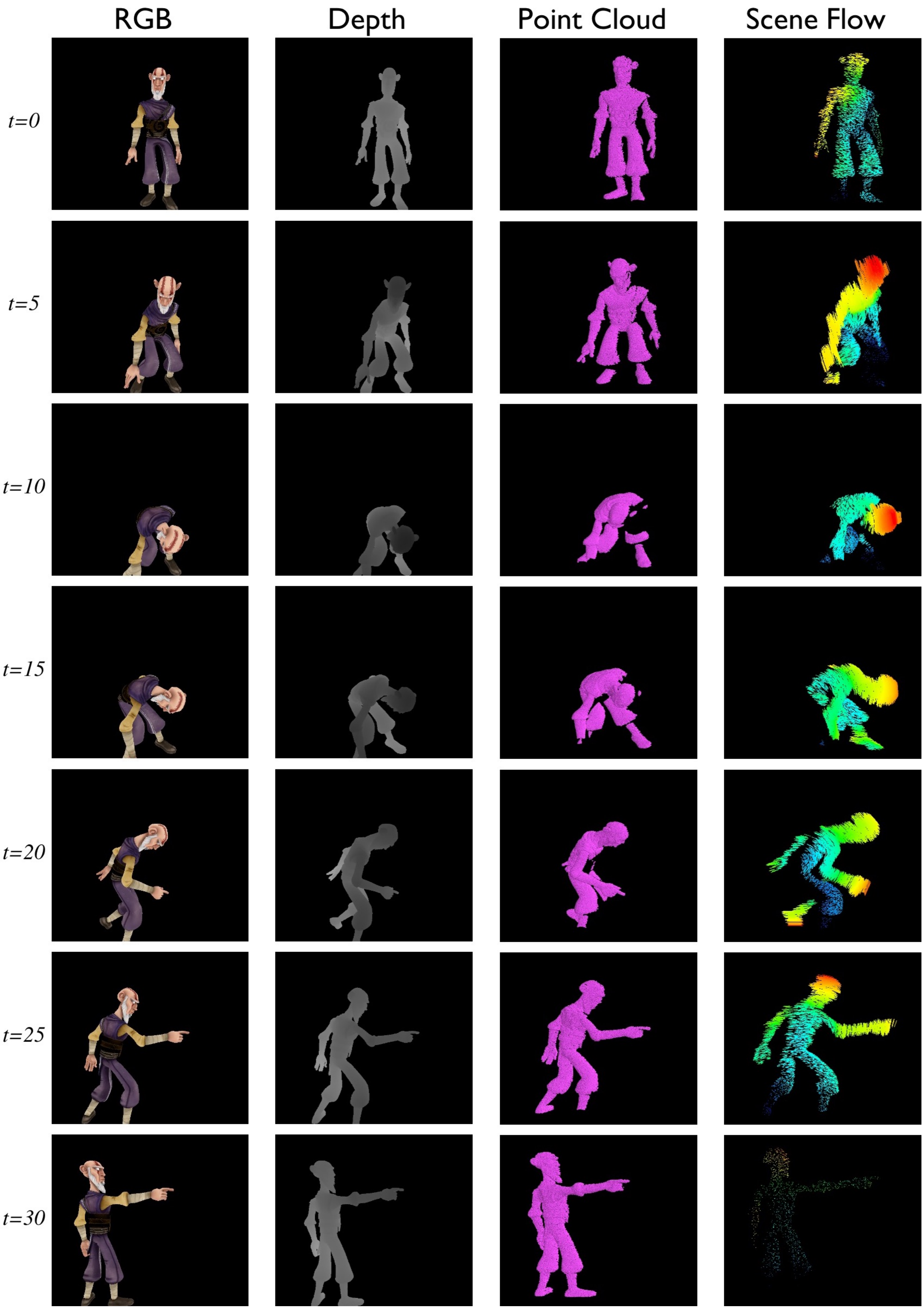


Figure 6.13 Rendered data for the Mixamo "Aiming Gun" sequence. From left to right: color image, depth image, point cloud (reporjected from depth image), and inter-frame scene flow. The visualization of scene flow and point cloud is done using Mayavi (https://docs.enthought.com/mayavi/mayavi/).

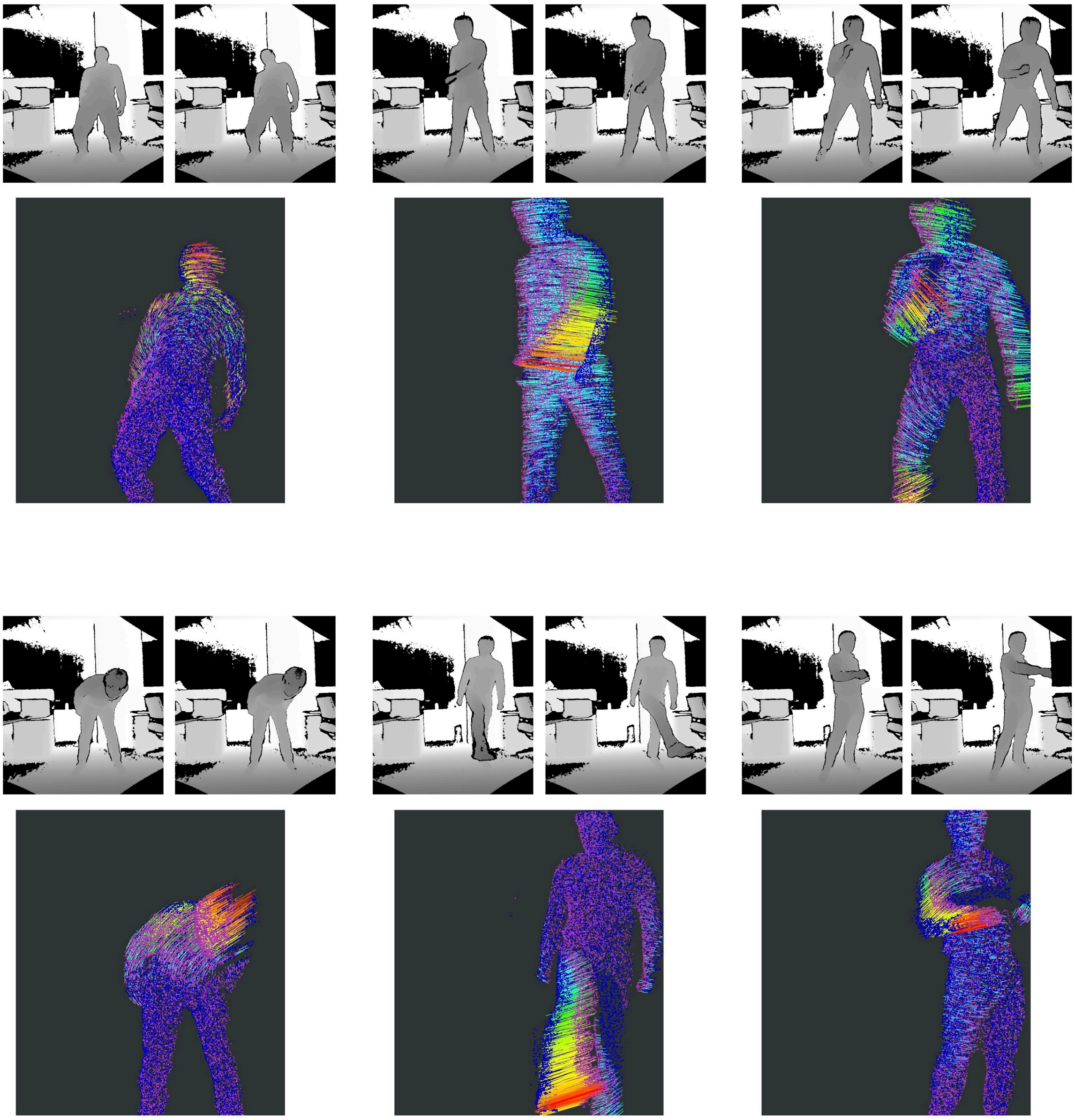


Figure 6.14 Scene flow estimation results for a real-world RGB-D sequence. The model is FlowNet3D [55] trained on synthetic dataset. The RBD-G sequence are captured using Azure Kinect. Upper row shows consecutive RGB-D input frames. Lower row show the scene flow vectors between the two frames, the point cloud sampled from source frame (given in pink color), and the point cloud sampled from the target frame (given in blue color). The visualization is done using Mayavi (https://docs.enthought.com/mayavi/mayavi/).

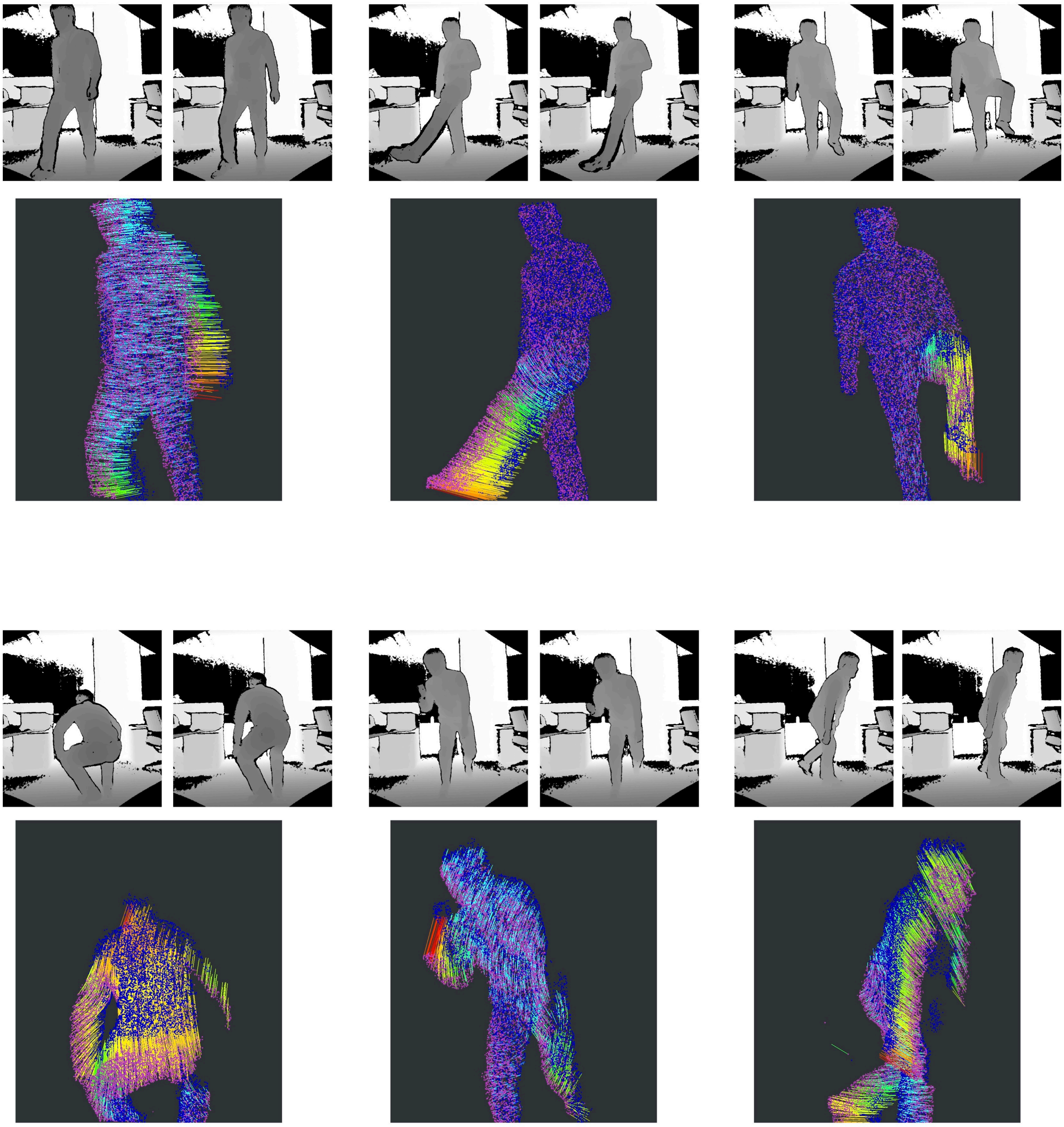


Figure 6.15 Scene flow estimation results for a real-world RGB-D sequence. The model is FlowNet3D [55] trained on synthetic dataset. The RBD-G sequence are captured using Azure Kinect. Upper row shows consecutive RGB-D input frames. Lower row show the scene flow vectors between the two frames, the point cloud sampled from source frame (given in pink color), and the point cloud sampled from the target frame (given in blue color). The visualization is done using Mayavi (https://docs.enthought.com/mayavi/mayavi/).

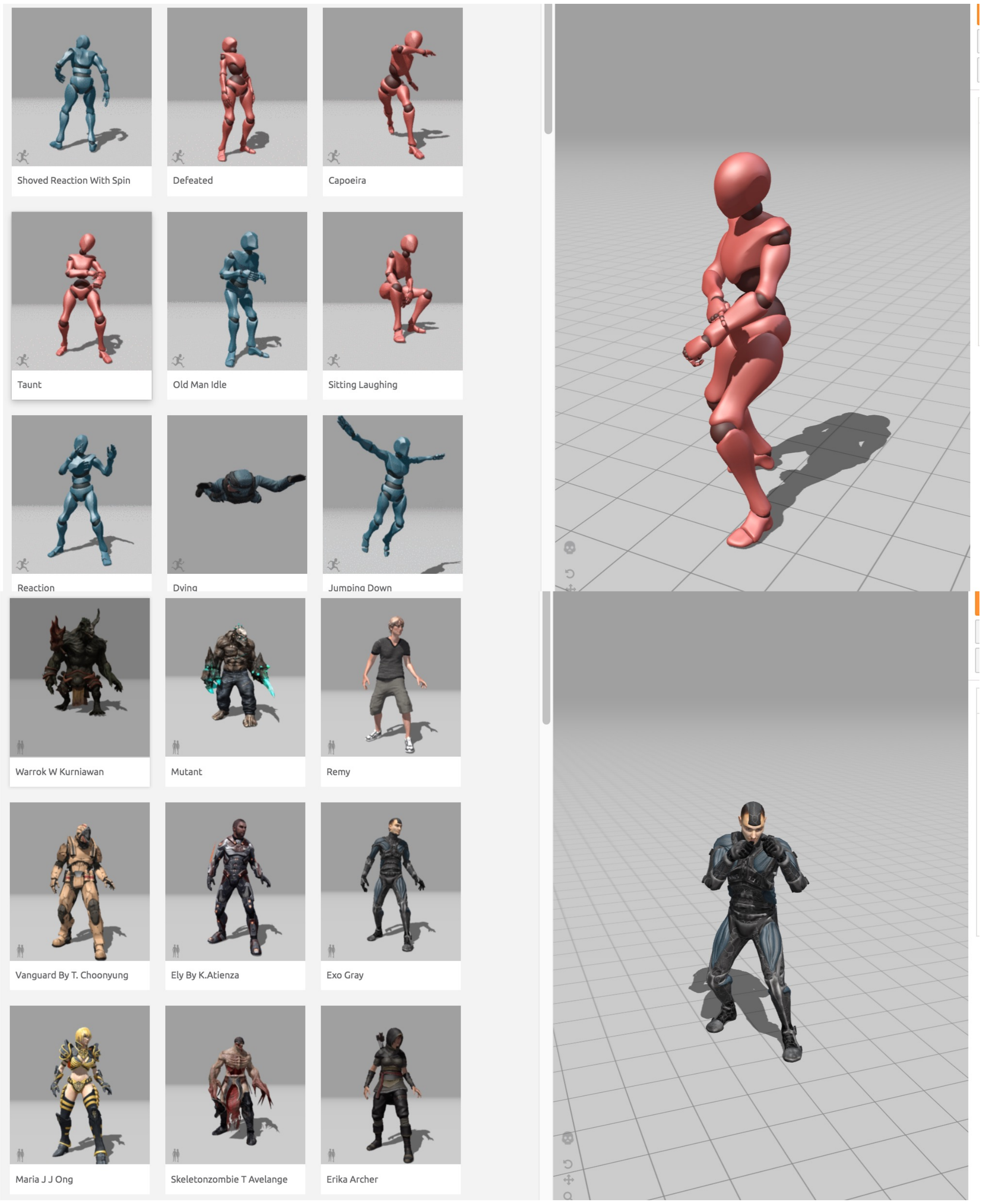


Figure 6.16 Example of animated characters in the Adobe Mixamo project (https://mixamo.com). Upper part of the figure shows the skeletal movement. Lower part of the figure shows the skin mesh. Generally, these characters are animated by using “rigging" and “skinning" to blend the skeletal movement to the surface skin mesh. The motion was collected using the motion capture system.

Chapter 7

Pose Graph Optimization for Unsupervised Monocular Visual Odometry

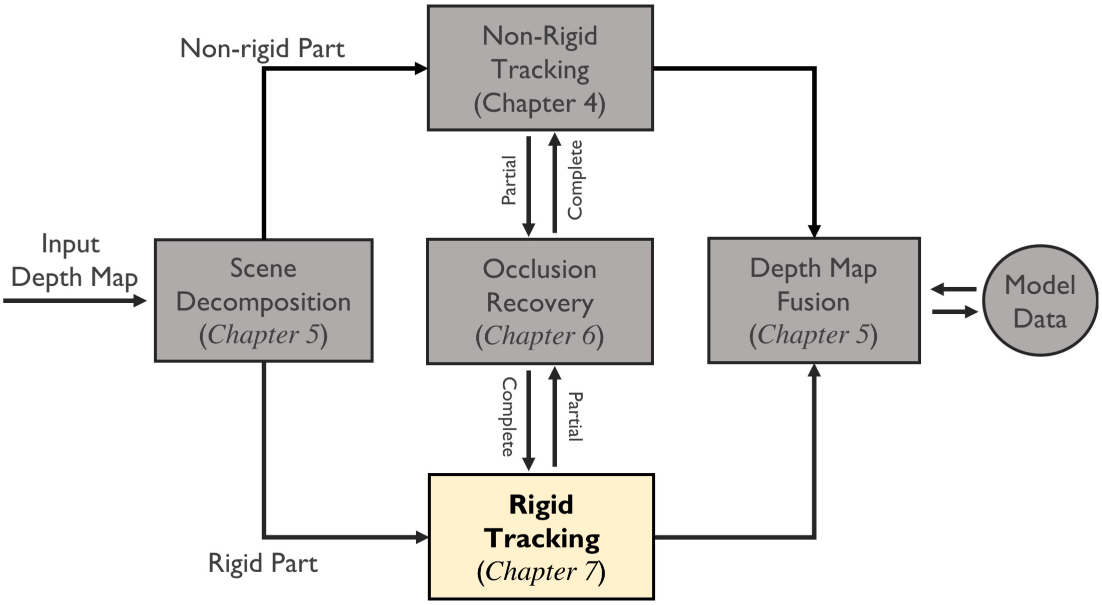

Non-rigid Part
Non-Rigid Tracking (Chapter 4)
Partial
Complete
Input Depth Map
Scene Decomposition (Chapter 5)
Occlusion Recovery (Chapter 6)
Depth Map Fusion (Chapter 5)
Model Data
Complete
Partial
Rigid Part
Rigid Tracking (Chapter 7)

Figure 7.1 The focus of this chapter and its position in the entire system.

As shown in Figure. 7.1, this chapter focuses on the rigid tracking part. Rigid tracking always suffers from trajectory drift when the camera travels a long distance. Camera trajectory drifts eventually cause geometry distortions in reconstruction. In this chapter, we pick out the monocular visual odometry (VO) problem and propose a method that corrects the trajectory drift. Note that such kind of technique is indispensable for large scale reconstruction.

Unsupervised Learning based monocular visual odometry has lately drawn significant attention for its potential in label-free leaning ability and robustness to camera parameters and environmental variations. However, partially due to the lack of drift correction technique, these methods are still by far less accurate than geometric approaches for large-scale odometry estimation. In this chapter, we propose to leverage graph optimization and loop closure detection to overcome limitations of unsupervised learning based monocular visual odometry. To this end, we propose a hybrid VO system which combines an unsupervised monocular VO called NeuralBundler with a pose graph optimization back-end. NeuralBundler is a neural network architecture that uses temporal and spatial photometric loss as main supervision and generates a windowed pose graph consists of multi-view 6DoF constraints. We propose a novel pose cycle consistency loss to relieve the tensions in the windowed pose graph, leading to improved performance and robustness. In the back-end, a global pose graph is built from local and loop 6DoF constraints estimated by NeuralBundler, and is optimized over $SE(3)$. Empirical evaluation on the KITTI odometry dataset demonstrates that 1) NeuralBundler achieves state-of-the-art performance on unsupervised monocular VO estimation, and 2) our whole approach can achieve efficient loop closing and show favorable overall translational accuracy compared to established monocular SLAM systems.

7.1 Introduction

Nowadays, monocular visual odometry (VO) can be newly divided into two groups based on the technique and the framework adopted: geometry-based and learning-based approaches. The geometric approach is usually solved via shallow feature or photometric re-projection followed by on-line error minimization. Learning base visual odometry is a newly emerged solution and has already achieved promising results on some benchmarks. This approach is solved by off-line training of End-to-End deep neural networks driven by a large number of image sequences. Benefiting from the nature of data-driven approach and the potential of deep neural networks, learning based monocular VO has shown clear advantages over geometric methods in the following aspects: 1) no need for parameter tuning effort, 2) robustness to tracking failure and scale drift [113], 3) capable of recovering metric scale from monocular image by using stereo image pairs in training phase [52][32][101], 4) high potential in directly integrating semantic information for robust camera tracking.

Due to the existence of outliers, noises, etc., all frame-frame VO systems suffer from drifts (the accumulation of small errors over time). In the geometric lineup, the so-called graph-based SLAM mitigates this problem by combining the geometric VO with an optimization back-end to continuously regulate the landmark's positions and the camera's poses, as known as the graph optimization technique. Graph-based SLAM has achieved success in many cases with established systems, such as feature-based PTAM [49] and ORB-SLAM [65], and direct LSD-SLAM [27] and DSO [28]. Especially, with the help of loop closing, i.e. graph optimization with correctly established loop closure constraints, SLAM is able to significantly reduce global trajectory drift.

Partially due to the lack of drift correction technique, learning based VO is still by far less accurate than the geometric approach. Therefore, it is worth exploiting the potential of SLAM's graph optimization technique for learning based visual odometry. However, such work has never been done. We presume that this is because most of the existing deep learning architectures are either trying to mimic the graph optimization process through memory-based LSTM network [95] or trying to integrate loop closing into a whole end-to-end learning process. These works ignore the fact that loop closing is randomly occurred event and requires simultaneously processing of sequence with arbitrary length, which the neural networks are inadequate to do.

In this work, we build on the main idea of the unsupervised VO architecture: SfMLeaner [113], the bag of visual word based place recognition tool: DBoW2 [29], and the insight of using Covisibility / Essential Graph optimization [49][65] for large-scale loop closing operation, to design a hybrid VO system with the following contributions:

- An unsupervised learning based monocular visual odometry called NeuralBundler which produces a windowed pose graph from monocular image sequence with a novel training loss that enforces pose cycle consistency.
- Efficient loop closing procedure based on the optimization of a pose graph which is built from local and loop 6DoF constraints estimated by the proposed unsupervised monocular VO.

We present the evaluation on KITTI odometry dataset [31]. NeuralBundler achieves the state of the art performance on learning based visual odometry estimation and our whole approach is able to perform efficient loop closing and yields favorable overall translational accuracy compared to established monocular SLAM systems. To the best of our knowledge, this is the first attempt to combine deep learning based VO with the classic graph optimization technique. Our research provides insights into the design of future SLAM system which could directly integrate the robustness and perception ability of deep neural network.

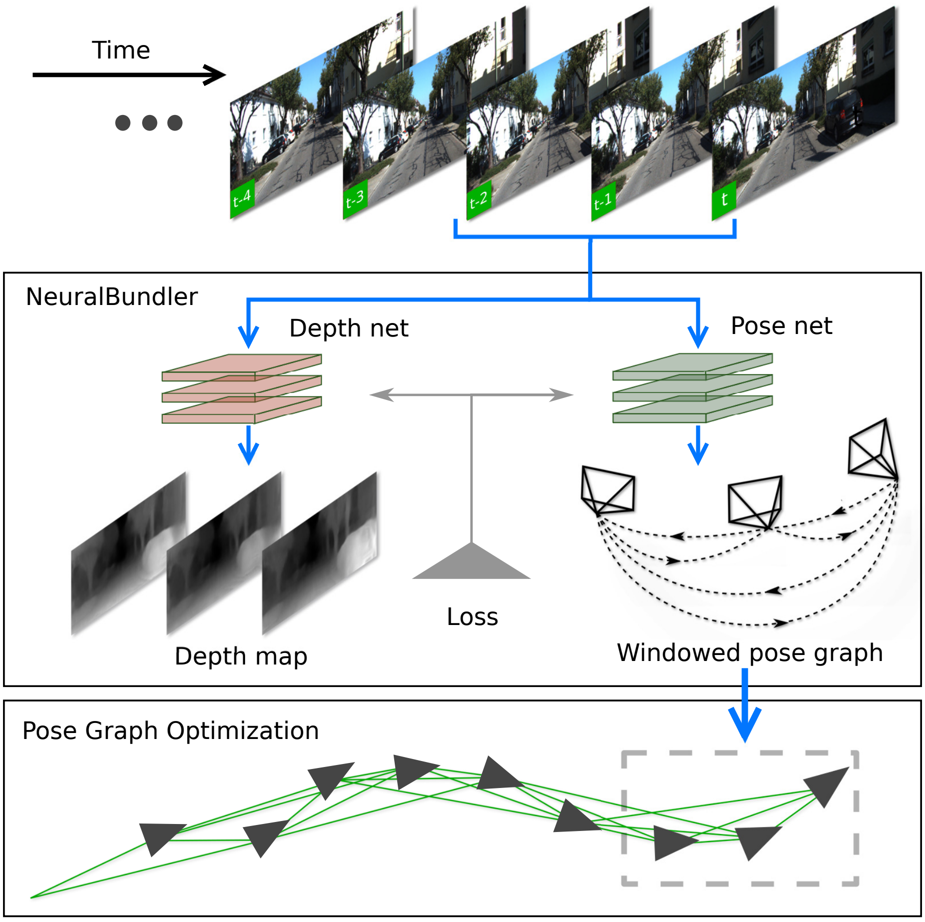


Figure 7.2 Overview of the proposed VO system.

7.2 Related work to visual odometry estimation

7.2.1 Geometry based visual vdometry

Geometry based Visual Odometry is a well-studied problem with two main solutions: feature-based and direct methods. Feature-based approach usually consists of two stages: 1) data association, through hand-engineered feature extraction (e.g. SURF [4], SIFT [59] and ORB [76]) and matching, and 2) pose estimation via minimizing feature point re-projection error. Direct VO treats data association and pose estimation as a whole optimization problem and solve it by minimizing the photometric error. Although these methods are effective in many cases, they are usually hard-coded and require extensive parameter tuning effort in order to ensure performance in a given scenario. The reliance on accurate data association can lead to tracking failure in regions of low texture, illumination change, and occlusions.

Moreover, geometric approaches inherently suffer from the fact that camera motion can only be estimated up to an unknown scale which also leads to scale drift over time.

7.2.2 Learning based visual odometry

This line of research can be further divided into 2 categories: supervised and unsupervised methods. In supervised approaches, Wang et al. [95] train a deep recurrent network end-to-end to predict ego-motion using ground truth trajectory as supervision. Kendall et al. [47] directly regress the camera's world pose from RGB images with the convolutional neural network. Zhou et al. [112] proposed a coarse to fine deep learning framework to track camera based on key-frame. However, the cost of collecting ground truth poses limits the application of such methods.

On the other hand, the unsupervised approach attracts more attention for its label-free leaning ability. The first architecture that achieved unsupervised learning of ego-motion from the video is SfMLeaner proposed by Zhou et al [113]. SfMLeaner takes consecutive temporal images to predict both depth and ego-motion with view synthesis as supervision. However, similar to geometric approaches, SfMLeaner can only observe ego-motion in a relative scale from monocular image. Godard et al. [32] show that the solution to recovery metric scale for depth prediction is using stereo images constraints for network training. Soon after, Nan et al. [101] integrate such a monocular deep depth predictor into DSO [28] as direct virtual stereo measurements. The follow-up works from Ruihao et al. [52] and Huangying et al. [107] also show that leveraging stereo image pairs with known baseline in training phase enable the networks to recover metric scale for both depth and pose estimation. Existing learning based methods only focus on frame-frame VO estimation. We propose to utilize the multi-view discrepancies within a temporal window. Our method produces a windowed pose graph and uses a novel loss to ensure pose consistency in the graph.

7.2.3 Graph-based SLAM

Graph-based SLAM maintains a global graph whose nodes represent camera's poses or landmarks and an edge represents a sensor measurement that constrains the connected poses [36]. Apparently, such constraints can be conflicting to each other since measurements are easily influenced by noise. Once such a graph is constructed, SLAM uses graph optimization method (i.e. nonlinear least-squares error minimization via the Gauss-Newton or Levenberg-Marquardt algorithm) to find a configuration of the nodes that is maximally consistent with all the constraints. The graph optimization procedure, with the presence of both camera pose and landmarks in the graph, is called Bundle-Adjustment (BA). Monocular SLAM systems that apply graph optimization include PTAM [49], LSD-SLAM [27], ORB-SLAM [65], DSO [28], etc.

In monocular SLAM, loop closure is solved through a pose graph optimization with 7DoF similarity constraints $(\text{Sim}(3))$ to correct the scale drift. A pose graph is built on selected key-frame connected by the pose-pose constraints. Pose-pose constraints are defined by covisiblity and estimated by a geometry base VO front end. Two poses are connected to each other if they share enough common features. The graph built on covisiblity is called Covisiblity Graph. In order to achieve scalable, real-time performance, Raúl et al. [65] proposed to perform loop closing on a much lighter Essential Graph which retains all the nodes (key-frames) from Covisibility Graph and a subset of edges with high covisibility. As shown in [65], the optimization of a properly constructed Essential Graph is already very accurate that full Bundle Adjustment only makes marginal improvement. We take the idea of loop closing with graph optimization and apply it to an unsupervised learning based visual odometry. Different from the monocular SLAM approach, we only optimize the pose graph with 6DoF constraints, i.e. $\text{SE}(3)$. The reason is that perhaps owing to the nature of data-driven method, in our learning based VO, scale drift is so small, that, in the experiments, optimization over $\text{SE}(3)$ and $\text{Sim}(3)$ almost leads to the identical result.

Loop closure is triggered by the place recognition technique. Appearance or image-image matching based methods, such as the bag of word approaches DBoW2 [29], are dominating this area for their high efficiency. Raúl Mur-Artal et al. [65] proposed a bag of words place recognizer built on DBoW2 with ORB feature and successful achieved real-time loop closing. In this work, for efficiency and simplicity, we used a similar loop closure detection procedure.

7.3 Method

7.3.1 Unsupervised monocular visual odometry

In this section, we will introduce our unsupervised approach for monocular visual estimation. The training procedure is shown in Fig. 7.3.

Network Architecture

We name our model NeuralBundler in the sense that, similar to Bundle Adjustment, it performs jointly optimization of both poses and 3D position (depth map) in a neural network fashion. NeuralBundler consists of a pose estimation network and a depth estimation network, which are trained jointly and can be used separately in the testing phase. The input to the pose estimation network is a stack of views from a sliding window of size N, and the

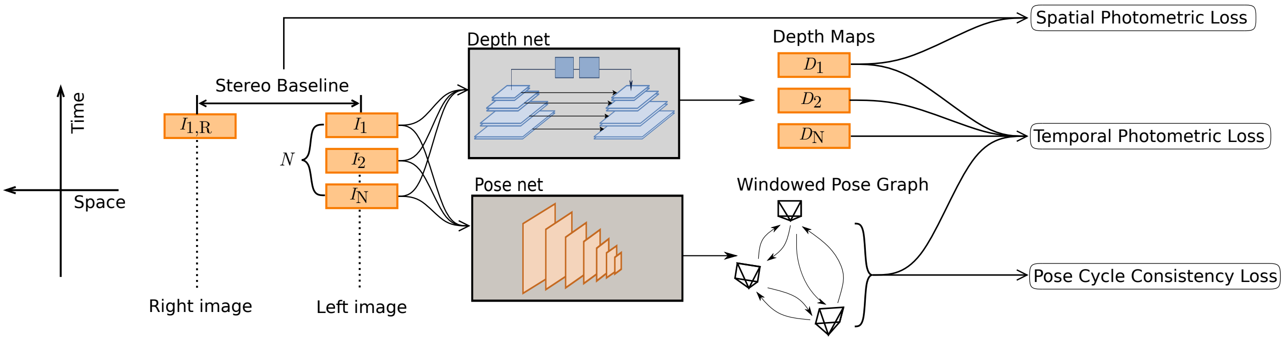


Figure 7.3 Training procedure of NeuralBundler. 1 right image from the first frame and N left images are involved in the training phase. Both pose net and depth net only require monocular sequence in the testing phase.

output is a windowed pose graph which has N nodes representing the views and $N \times (N-1)$ edges each of which represents the relative 6DoF motion between two views. The network consists of 7 stride-2 convolutions followed by two 1×1 convolutions with $6 \times N \times (N-1)$ output channels (corresponding to 3 Euler angles and a 3D translation for each edge in the windowed pose graph). Depth net produces one dense depth map for each RGB image. It has an encoder-decoder shape with skip connections between corresponding encoder and decoder blocks in order to generate high-resolution depth prediction with fine-grained details. The encoder is based on ResNet-50 and each decoder block uses a nearest-neighbor upsampling layer followed by a convolutional layer.

Loss Function

Let's denote N as the total number of views in the input window, I_i as the i-th view. D_i denotes the predicted depth map and T_{ij} denotes the predicted relative motion from view i to view j. K is the camera's intrinsics. $\mathbb{E}$ is the set of the graph's edges, each of which is represented by a tuple of index: (i, j). The final loss is a weighted sum of the photometric loss and pose cycle consistency loss.

Temporal Photometric Loss

Let p_i be the homogeneous coordinates of a pixel in view i and p_j as p_i's projected pixel onto the view j. Based on the epipolar geometry, we can obtain p_j from p_i through:

$$p_j = KT_{ij}D_iK^{-1}p_i$$

Then, by applying the differentiable bilinear sampling mechanism proposed in spatial transformer networks [43], from view i we can synthesis view j, which is denoted as $I_{i\to j}$.

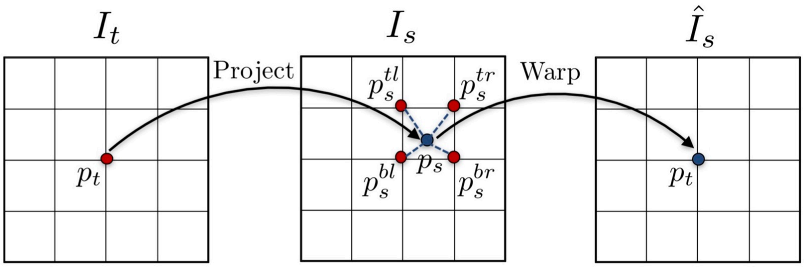


Figure 7.4 Illustration of the differentiable image warping process. For each point p_t in the target view, we first project it onto the source view based on the predicted depth and camera pose, and then use bilinear interpolation to obtain the value of the warped image $\hat{I}_s$ at location p_t. This process is made differentiable by spatial transformer networks [43].

Inspired by Godard et al. [32], the quality of the reconstructed image is measured with the weighted sum of the l_1 loss and the single scale structural similarity (SSIM) loss. Then the temporal photometric loss is:

$$L_{pho}^{temp} = \sum_{(i,j)\in\mathbb{E}} \left[(1-\alpha)L^{l_1}(I_j, I_{i\to j}) + \alpha L^{SSIM}(I_j, I_{i\to j})\right]$$

where α is set to 0.25.

Stereo Spatial Photometric Loss

In order to recover metric scale for depth and pose estimation, like [107], we apply the same photometric loss between the stereo camera. As show in Fig. 7.5, the warpping function between a pair of stereo images can be simply represented by the *disparity*, which represent the displacement of pixel in the image's horizontal axis.

As shown in Fig. 7.3, we only use the stereo images from the beginning frame of the window. Therefore I_1 (default as left image) and $I_{1,R}$ (right image) is the spatial pair. The projection $I_{1,R\to 1}$ is synthesized using the disparity. As shown in Fig. 7.5, the disparity can be computed from the known stereo baseline and the predicted left depth map. As shown in Fig. 7.6, the stereo images are undistorted and rectified, image synthesized can be done with disparity. Then the spatial stereo photometric loss is:

$$L_{pho}^{spat} = (1-\alpha)L^{l_1}(I_1, I_{1,R\to 1}) + \alpha L^{SSIM}(I_1, I_{1,R\to 1})$$

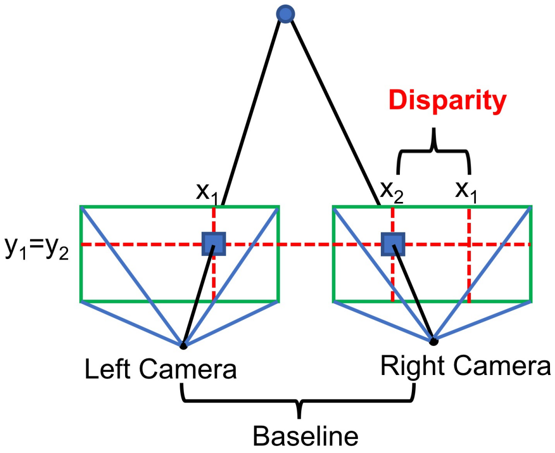


Figure 7.5 Illustration of the stereo constraints. Since the baseline (horizonta distance of two camera) and the camera focal length is known. The disparity-to depth relation can be represented by $disparity = \frac{baseline \times focallength}{depth}$

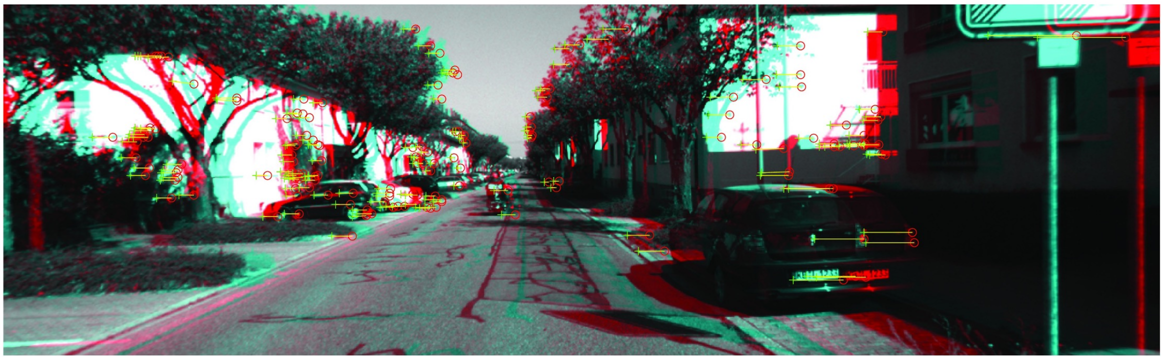


Figure 7.6 Stereo images overlayed from KITTI dataset. Since the stereo images are undistorted and rectified, the pixels matches are along parallel (horizontal) lines. Therefore, the disparity can be used to warp a image.

Pose Cycle Consistency Loss

The windowed pose graph is a complete directed graph containing $N \times (N-1)$ 6DoF camera motion constraints. Obviously, these constraints may be contradictory to each other, which creates tensions in the pose graph. During network training, we relax the windowed pose graph by penalizing a pose cycle consistency loss. Say (i, j, k) are the indexes of three views in the input window. Then the cycle constraint $T_{ij}T_{jk}T_{ki} = \mathbf{I}$ holds, where $\mathbf{I}$ is the identity matrix. Let's denote $\mathbb{C}$ as the set of possible cycles in the graph. We penalize the l_1 loss:

$$L_{pos} = \sum_{(i,j,k) \in \mathbb{C}} L^{l_1}(T_{ij}T_{jk}T_{ki}, \mathbf{I})$$

Considering that this loss is similar to the objective of graph optimization (in Section IV-B), we are somewhat performing a windowed pose graph optimization in a neural network form. Section V-B demonstrates the efficiency of this loss.

7.3.2 Back-end

In the back-end, as shown in Fig. 1, we maintain a global pose graph and insert new elements to it each time a new frame is processed by NeuralBundler. We optimize this graph if a loop is established.

Pose graph construction

Pose graph is built on local and loop pose-pose constraints estimated by NeuralBundler. As shown in Fig. 3, local constraints are generated in a sliding window. Loop constraints are obtained in two crossed windows around the loop closure area. Specifically, when there is a loop closure detected between views I_i and I_j, the two input windows for NeuralBundler are $< I_i, I_{j-1}, ..., I_{j-N+1} >$ and $< I_j, I_{i-1}, ..., I_{i-N+1} >$, where N is the window size.

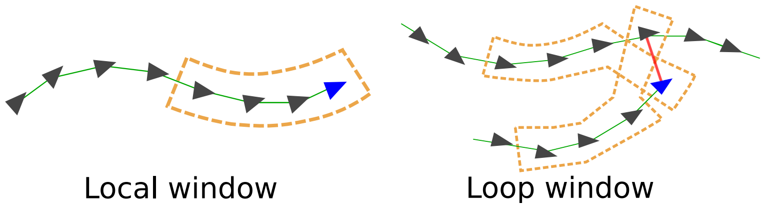


Figure 7.7 Image windows for building local and loop pose-pose constraints. The red link indicates a detected loop. Live frames are marked with blue color.

The system uses a bags of visual words place recognition tool, based on DBoW2, to perform loop closure detection. We use a predefined ORB-Vocabulary and ORB-Database which are created off-line with ORB descriptor extracted from a large set of images. Since querying the database with an image will return multiple candidates, similar to [65], we apply the following procedure to filter the candidates: in order to be accepted, 1) a loop candidate must be preceded by 6 or more consecutive loop detections, and 2) the loop candidate's corresponding 6DoF transformation must get enough in-liers after several RANSAC iterations. Fig. 7.8 shows a loop closure example on the Seq-06 in the KIITI odometry dataset [31].

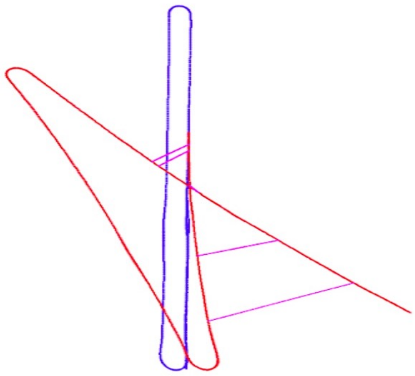


Figure 7.8 Loop closure detection example on the Seq-06 in the KIITI odometry dataset [31]. The red trajectory show the estimated trajectory. The blue one isthe ground truth trajectory. the pink links indicate detected loops using the DBoW2 [29] tool.

Pose Graph Optimization

A 3D rigid body transformation $\mathbf{T} \in \mathrm{SE}(3)$, is defined by:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \text{ with } \mathbf{R} \in \mathrm{SO}(3) \text{ and } \mathbf{t} \in \mathbb{R}^3$$

During optimization, $\mathbf{T}$ is mapped to a minimal representation in $\mathbb{R}^6$ of the associated Lie-algebra through the logarithmic mapping function $\log_{\mathrm{SE}(3)}$. Given a pose graph constructed in the proposed way, the error in an edge is defined as:

$$\mathbf{e}_{i,j} = \log_{\mathrm{SE}(3)}(\mathbf{T}_{ij}^{-1}\mathbf{T}_i^{-1}\mathbf{T}_j)$$

where $\mathbf{T}_{ij}$ is the relative 6DoF transformation constraints, and the goal is to minimize the total energy:

$$\chi^2(\mathbf{T}_2, ..., \mathbf{T}_m) = \sum_{\mathbf{T}_{ij}} (\mathbf{e}_{i,j}^T \mathbf{e}_{i,j})$$

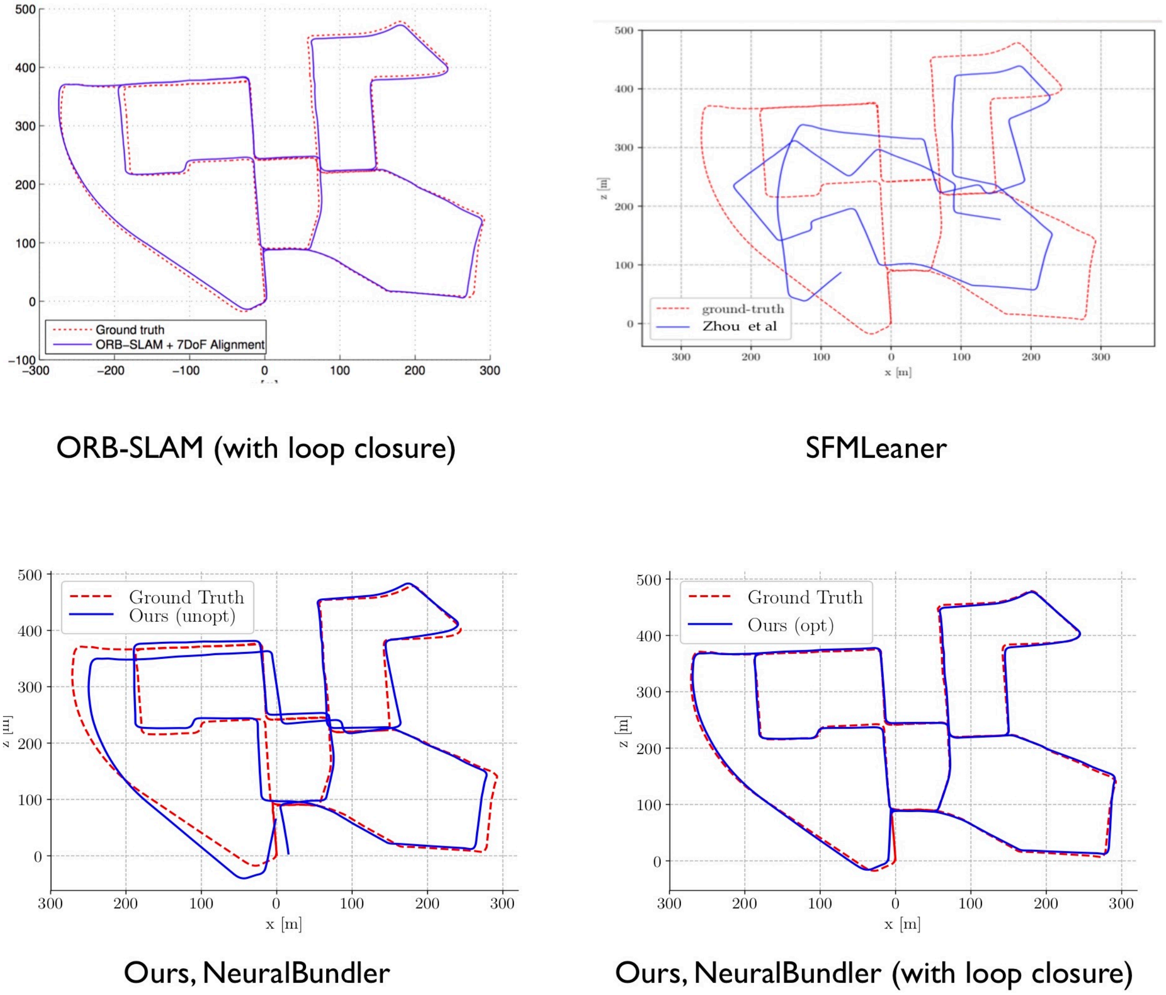
with respect to the absolute poses $\mathbf{T}_2, ..., \mathbf{T}_m$. These absolute poses are initialized through a chain of relative 6DoF transformations starting from the world reference frame $\mathbf{T}_1$, which is fixed during the optimization. We use the Levenberg-Marquardt algorithm implemented in g2o to carry out the optimization.

Approach		00	01	02	03	04	05	06	07	08	09	10
UnDeepVO [52]	t_{rel}	4.41	–	5.58	–	–	3.4	–	**3.15**	4.08	–	–
	r_{rel}	1.92	–	2.44	–	–	1.5	–	2.48	1.79	–	–
SfMLeaner [113]	t_{rel}	60.53	35.17	58.75	11.0	4.49	19.14	25.88	21.33	21.90	54.38	14.33
	r_{rel}	6.12	2.72	3.56	3.94	5.24	4.11	4.81	6.65	2.91	3.21	3.30
Huangying et al. [107]	t_{rel}	6.62	23.98	6.93	–	3.01	5.12	5.92	7.06	5.85	12.16	13.00
	r_{rel}	3.57	1.78	2.39	–	2.02	2.43	2.10	3.8	2.54	3.61	3.54
Ours, NB (*no pccl*)	t_{rel}	10.31	45.45	7.32	7.97	3.5	5.80	4.64	6.70	5.54	11.98	**12.28**
	r_{rel}	2.78	2.13	2.71	4.16	2.01	2.26	3.85	3.46	2.10	3.43	3.97
Ours, NB	t_{rel}	4.33	**17.98**	6.89	**4.51**	**2.3**	3.91	4.6	3.56	**4.04**	8.10	12.9
	r_{rel}	1.85	**1.44**	2.61	**2.82**	**0.87**	1.64	2.85	2.39	**1.53**	2.81	**3.17**
Ours, NB+*LC*	t_{rel}	**3.24**	–	**4.85**	–	–	**1.83**	**2.74**	3.53	–	**6.23**	–
	r_{rel}	**1.35**	–	**1.60**	–	–	**0.7**	**2.6**	**2.02**	–	**2.11**	–

Table 7.1 Comparison with unsupervised learning based approaches. $t_{rel}(\%)$ is translational error and $r_{rel}(^o)$ is rotational error. Both are averaged over 100m to 800m intervals. Result of UndeepVO is obtained from [52] and for SfMLearner [113] and Huangying et al. [107] we ran their pre-trained model. All models share the same training setup. Three variants of our approach is included, NB (*no pccl*): pose cycle consistency loss is not used for training, NB: NeuralBundler with pose cycle consistency is used for training, NB+*LC*: NeuralBundler+loop closing, our whole approach. Both the pose cycle consistency loss and loop closing show improvement over the baseline.

Approach	RMSE (m)										
	00*	02*	05*	06*	07*	09*	01	03	04	08	10
Ours, NeuralBundler + Opt	**4.36**	52.75	**2.57**	**3.71**	**1.67**	14.85	**57.4**	**1.43**	**0.82**	**16.4**	11.2
ORB-SLAM	6.68	21.75	8.23	14.68	3.36	7.62	X	1.59	1.79	46.58	**8.68**
ORB-SLAM + Global BA (20 its.)	5.33	**21.28**	4.85	12.34	2.26	**6.62**	X	1.51	1.62	46.68	8.80

Table 7.2 RMSE error of estimated trajectories on KITTI Odometry Dataset. All the methods listed in the table perform loop closing if any loop is detected. Results of ORB-SLAM and ORB-SLAM + Global BA (20 its.) are taken from [65]. ORB-SLAM: perform optimization on the Essential Graph [65], ORB-SLAM + Global BA (20 its): perform 20 iterations of global Bundler Adjustment afterward. For trajectory alignment, we use 6DoF transformations and ORB-SLAM uses 7DoF transformations. Transformations are optimized to achieve the best alignment.



ORB-SLAM (with loop closure)

SFMLeaner

Ours, NeuralBundler

Ours, NeuralBundler (with loop closure)

Figure 7.9 Results on sequence 00 from the KITII Odometry dataset. Compared to SFM-Leaner [113], NeuralBundler yield better trajectory. After loop closing, our method show similar result with ORB-SLAM [65].

7.4 Experimental results

Implementation Detail

We implemented the networks using the publicly available TensorFlow framework and train it with Tesla P100 GPUs. We trained our model from scratch for 30 epochs, with a mini-batch size of 4 using Adam optimizer [48], where $\beta_1 = 0.9$, $\beta_2 = 0.999$. We used an initial learning rate of 0.0001 and halve it every 1/5 of the total iterations. The size of the image window for the windowed pose graph estimation network is set to 3 and each image is resized to 416×128. In the testing phase, both the network inference and graph optimization are carried out on an Ubuntu PC equipped with GeForce GTX TITAN X GPU and Intel Core i5 2.4 GHz

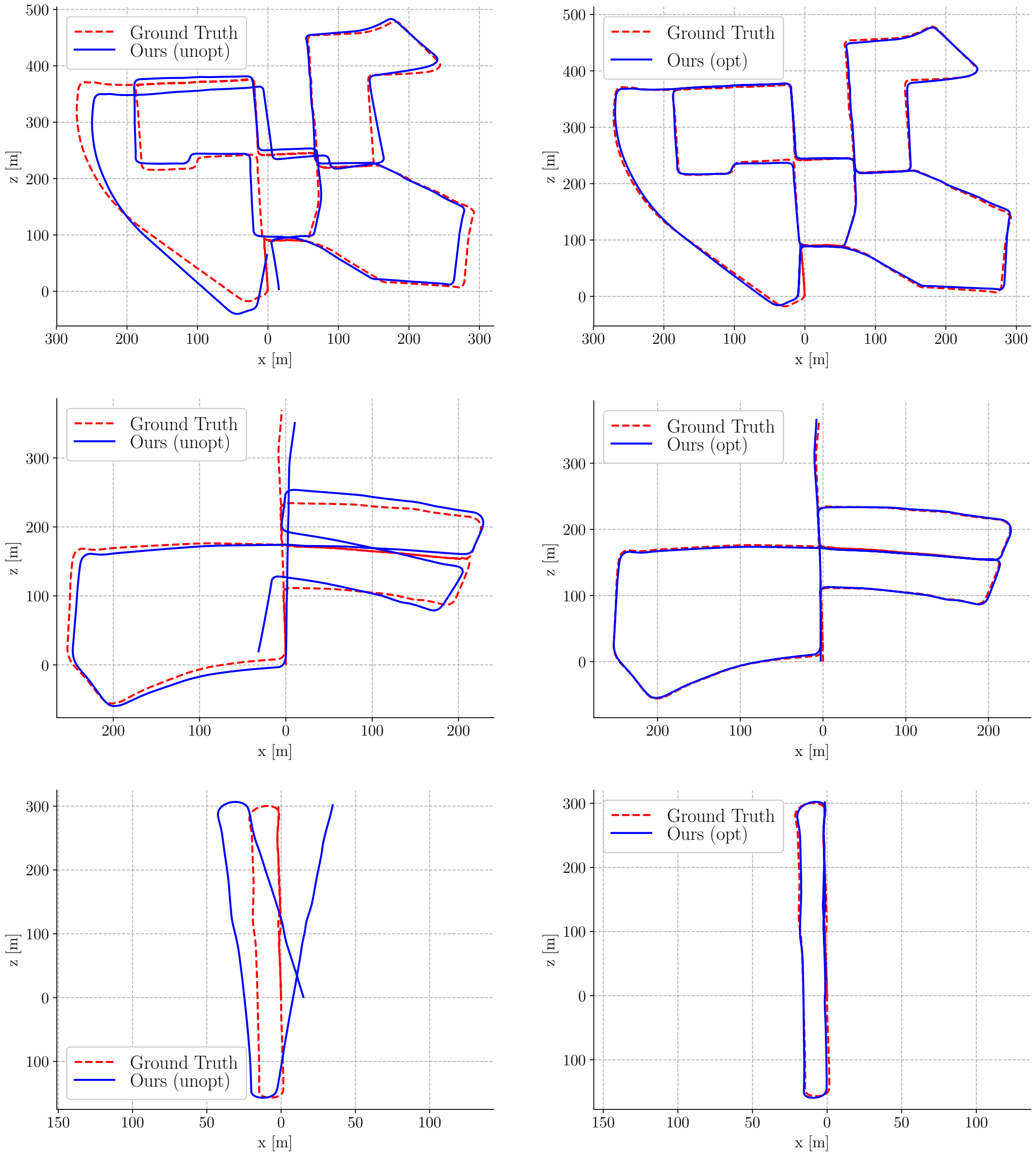


Figure 7.10 Results on sequence 00, 05 and 07 from the KITII Odometry dataset. Top row: Raw estimation of NeuralBundler (Trajectory is constructed using the inter-frame motion: $T_{1\to 0}$ from the windowed pose graph, see Section III). Bottom row: After performing loop closing (Graph optimization with loop constraints).

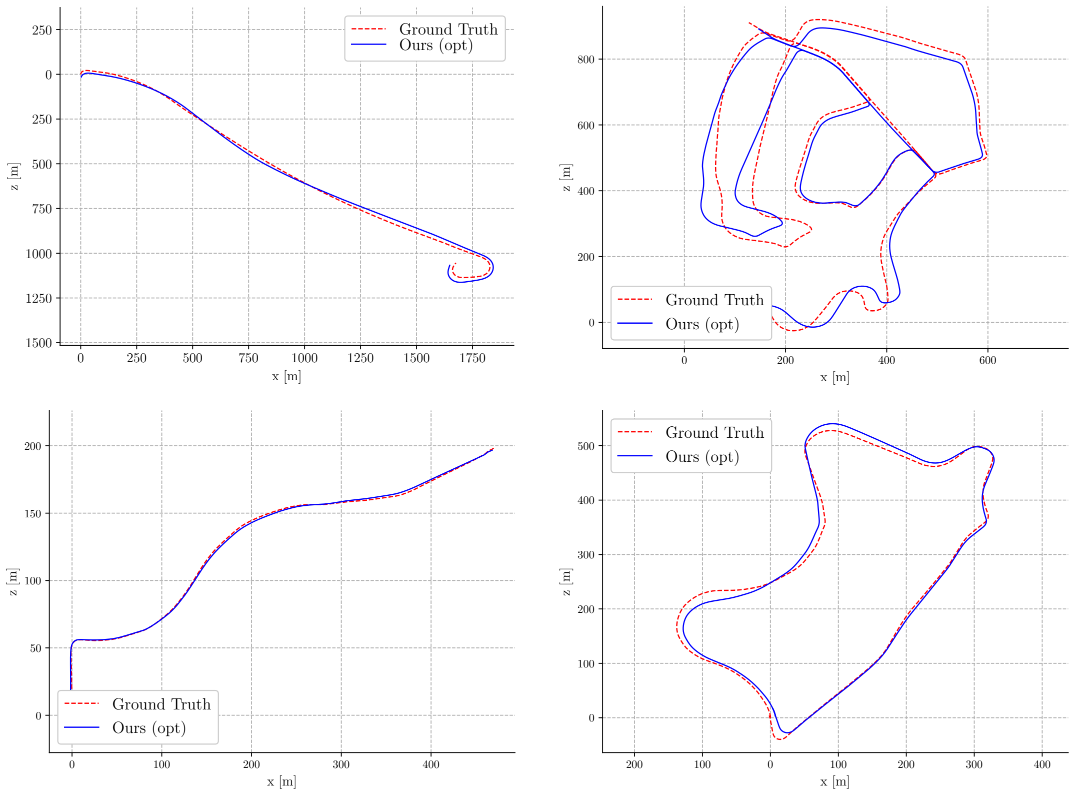


Figure 7.11 Results of our approach on sequence 01, 02, 03, and 09 from the KITII Odometry dataset.

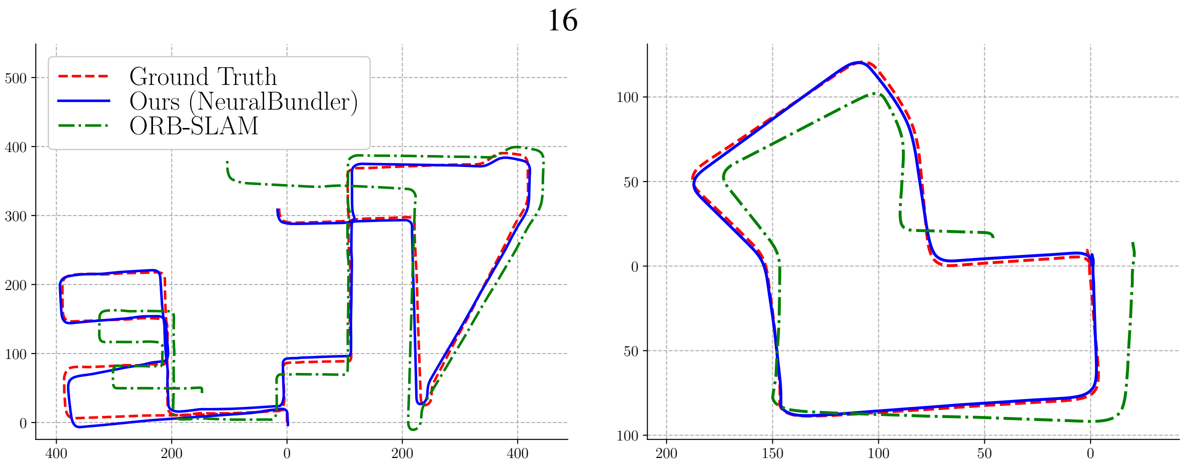


Figure 7.12 Results on sequence 08 (left) and 07 (right). Sequence 08 does not contain a loop. For sequence 07, we turn off the loop closing thread of ORB-SLAM. ORB-SLAM suffers from severe scale drift, and heavily rely on loop closing to eliminate scale drift.

CPU. The pose estimation network, with about 168k parameters, can be used separately in testing time. Pose estimation network requires less than 400MB GPU memory with over 40 Hz real-time performance. For benchmarking, we apply the KIITI odometry datasets [31] which contain 11 sequences (00-10) with ground truth trajectory obtained through the IMU/GPS readings. Fig. 7.13 show the details of this dataset.

Evaluation

Qualitative comparisons of our trajectories and the ground truth are shown in Fig. 7.11, Fig. 7.12 and Fig. 7.10. We align the trajectories with the ground truth through rigid body transformations on SE(3) (No scaling is applied for that our VO system can recover the metric scale). Sequences 00, 02, 05, 06, 07, 09 contain loops that were correctly detected and closed by our system. Fig. 7.10 shows the effectiveness of our loop closing approach on sequence 00, 05 and 06. The top row shows the raw VO estimation of NeuralBundler, and the bottom row shows corresponding results after performing loop closing. The raw VO estimation gives decent local accuracy while with gradually accumulated drift. After performing graph optimization with correctly detected loops, the drift is significantly reduced, leading to good fits with the ground truth.

For evaluation, first, we use the error metrics proposed in [31] to compare our approach with other unsupervised learning based visual odometry systems: UnDeepVO [52], SfM-Leaner [113] and Huangying et al. [107]. As shown in table I. For ablation study, our NeuralBundler consistently yields better results than the variant without pose cycle consistency loss. NeuralBundler outperforms SfMLeaner and Huangying et al. with a large margin and shows comparable results to UnDeepVO (which trains the network with extra supervisions from point cloud alignment and stereo camera's left-right pose consistency). Applying loop closing significantly reduced the error, while graph optimization without loop constraints (not shown) does not necessarily lead to better results, which makes clear the need of loop closures for global drift correction.

Then, we compare the results of our method with the state of the art feature-based ORB-SLAM. This time, we directly assess the overall translational RMSE error of the trajectory. As shown in table II, our method achieves smaller overall translational error for most of the sequences (8 out of 11). This makes sense considering that though both methods use monocular image, our approach is more robust to scale drift. Fig. 7.12 shows the estimated trajectory on sequence 07 and 08. Geometry based ORB-SLAM suffers from sever scale drift, while NeuralBundler achieves superior performance on eliminating the scale drift. ORB-SLAM encounters tracking failure on sequence 01 (a high way sequence). However, partially due to the fact that geometric methods know exactly the pixel-pixel correspondence, ORB-SLAM yields smaller rotational errors, which could explain our 3 lost cases.

7.5 Summary

In this chapter, we have presented a new monocular VO system which has an unsupervised learning based front-end called NeuralBundler and a graph optimization back-end. Results on KITTI odometry have proved the effectiveness of pose cycle consistency loss. Our whole approach can achieve efficient loop closure and show better overall RMSE than ORB-SLAM for most sequences in KITTI odometry datasets.

7.6 Limitation

A number of major challenges are yet to be addressed:

- We have not applied any key-frame selection and edge removal procedure yet. In our method, every frame becomes a "key-frame", and each node is connected with 12 edges (in the case that window size *N* is 3 and the node is not on the head or tail), which lead to a much denser and more complex pose graph than the SLAM's version. Though real-time loop closing is achieved, such techniques are still necessary for the sake of scalability and efficiency.

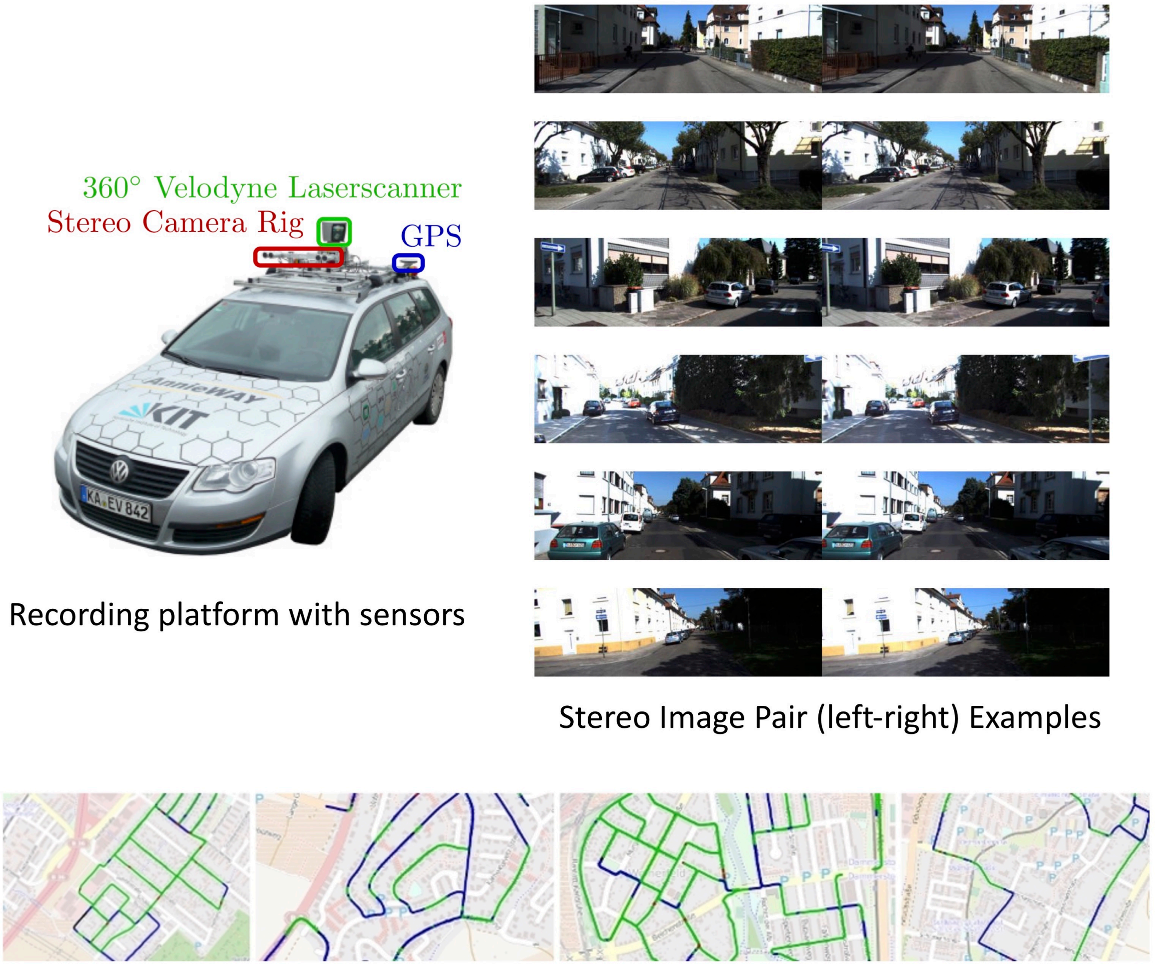
- The model is trained on sequences with fixed camera intrinsics, fixed input image size, and from limited scenes (car driving on the road). We need to solve the domain adaptation problem when applying to different situations.

- We still need to extract and maintain features in the background in order to use DBoW2 tool for loop closure detection, which considerably consumes memory and CPU power. The solution is to directly perform scene recognition with the deep neural network. We put this to future work.

- Unsupervised VO still lose to traditional methods on Mean Rotation/Translation accuracy. A possible solution is to use the coarse-to-fine strategy to optimize the VO estimation as in [112].

7.7 Supplementary material

In this document, we provide more details to the main chapter. Fig. 7.13 shows the details of KITTI odometry benchmark. The odometry benchmark consists of 22 stereo sequences. Among them, there are 11 sequences (00-10) with ground truth trajectories and 11 sequences (11-21) without ground truth.



Recording platform with sensors

Stereo Image Pair (left-right) Examples

Trajectory from visual odometry benchmark

Figure 7.13 Recording platform (left top), stereo image examples (right top), and trajectory from the KITTI odometry dataset [31]. The images of the car and trajectory maps are obtained from KITTI's official site (http://www.cvlibs.net/datasets/kitti/).

Chapter 8

System Demonstration

In this chapter we show a demonstration of our system that combines scene decomposition, non-rigid shape completion, rigid tracking and depth map fusion for a real-world sequence.

8.1 Implementation details

The hardwares are shown in Fig. 8.1 We record the demonstration sequence using Kinect 2 camera. Which connects to our desktop computer with USB 3.0 and needs extra charge cable. It provides depth resolution of 640×480. The RGB color image is calibrated to align with the depth image, where the RGB value of the pixels that does not have valid depth observation is set to (0,0,0). We run our system on the alienware area-51 computer, which contains two GeForce GTX TITAN graphics card and an Intel(R) Core(TM) i7-5930K CPU.

We implemented our system using a combination of python/PyTorch and c++/cuda, with wrappers in between two languages. Scene decomposition and non-rigid shape completion are running PyTorch models. Rigid tracking ICP program and depth map fusion are done in c++.

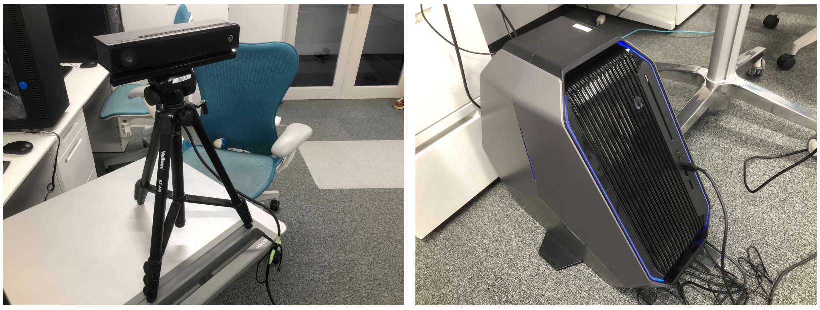


Figure 8.1 Left: the Kinect 2 sensor in this experiment. Right: the desktop computer for reconstruction.

8.2 Experiment results

We record a scene with a man walking around in a laboratory. The scene is captured via a Kinect 2 sensor which is manually moved by another person while scanning. The man occasionally appears and disappears from the camera view frustum. This sequence is extremely challenging because the non-rigid human motion is very fast meanwhile the camera is also moving.

Fig. 8.2 shows an example of foreground segmentation and the corresponding shape completion result. In this demonstration, for foreground segmentation, we use YOLO [7] to detect the human bonding box from the color image, the detailed segmentation is obtained via geometry post-processing on the point cloud.

Fig. 8.6 shows the reconstructed static background and the camera trajectory.

Fig. 8.3, 8.4, 8.5 show the screen shots of the scanning process of our method. Each screenshot captures the reconstructed scene, the input image, and the projection to the camera. Note that both the camera and the human are moving. Our system manages to reconstruct simultaneously both the environment and the non-rigid human.

Fig. 8.7 shows the fly-through video of the 4D reconstruction after the scanning process is finished, where we manually align the “two" person in the time axis.

Fig. 8.8 shows the ablation study of the background reconstruction. We show that both pose graph optimization (c.f. Chapter 7) and dynamic object segmentation (c.f. Chapter 5) are necessary for background reconstruction.

8.3 Limitation

We found that non-rigid object re-identification is a necessary yet interesting task. For example, at some point of the reconstruction, the human exit the view frustum of the camera, we do not know where it goes, thus we leave it as a static object in the scene (c.f. second screenshot in Fig. 8.4). A few seconds later, the same person re-enters the camera view frustum, we do not know who it is, thus regarding it as a new person (c.f. second screenshot in Fig. 8.5). We leave object re-identification as future work.

We also want to clarify that our system does not perform in real-time. Especially, due to the heavy 3D convolution operations, the non-rigid shape completion part runs at around 1 HZ on the GeForce GTX TITAN graphics card. We leave the real-time implementation as future work.

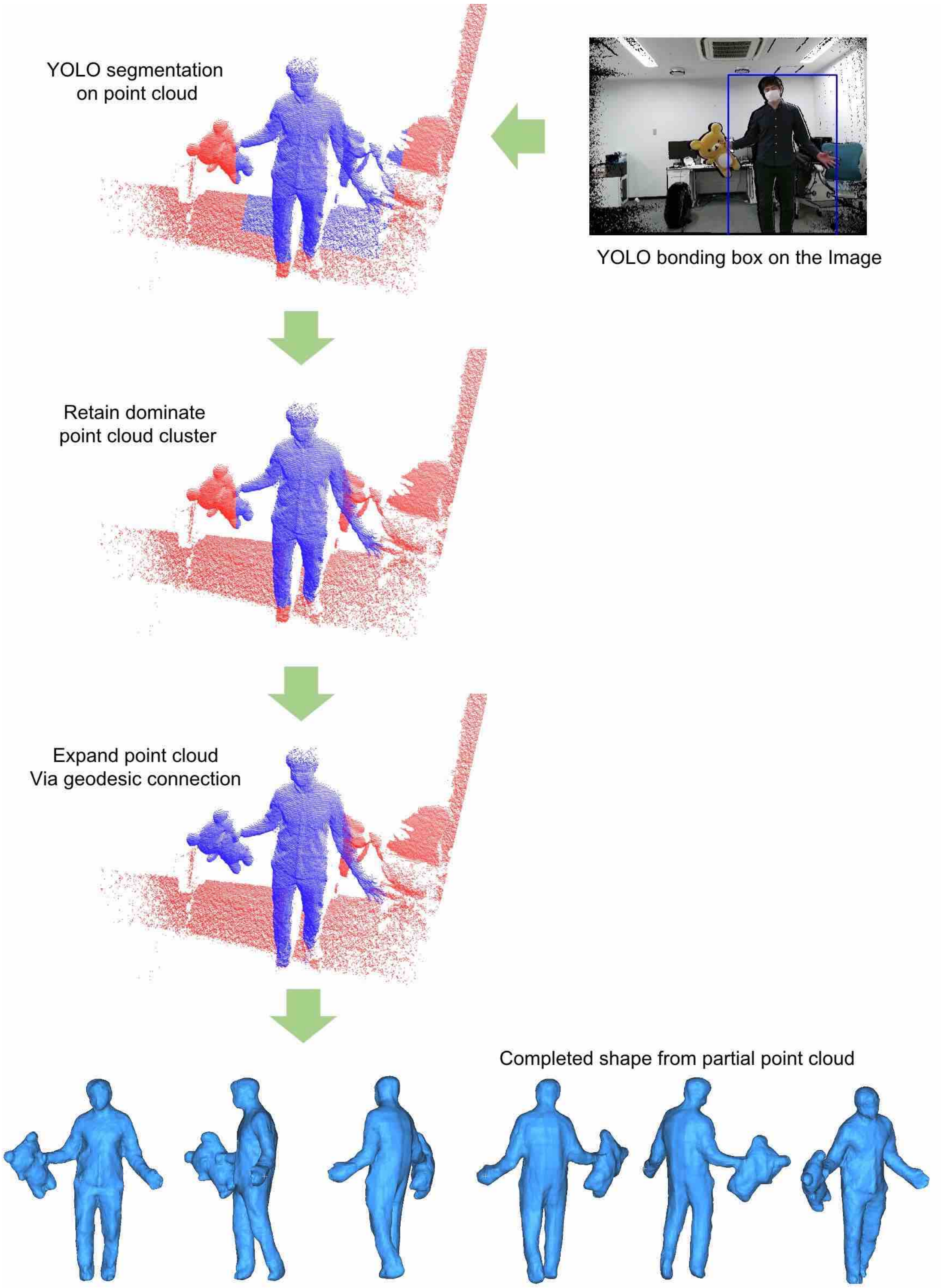


Figure 8.2 Foreground segmentation and completion for a RGB-D frame. In this demonstration, for foreground segmentation, we use YOLO [7] to detect the human bonding box from the color image, the detailed segmentation is obtained via geometry post-processing (i.e. point cloud clustering and geodesic expansion). The lasst row shows the completed foreground human.

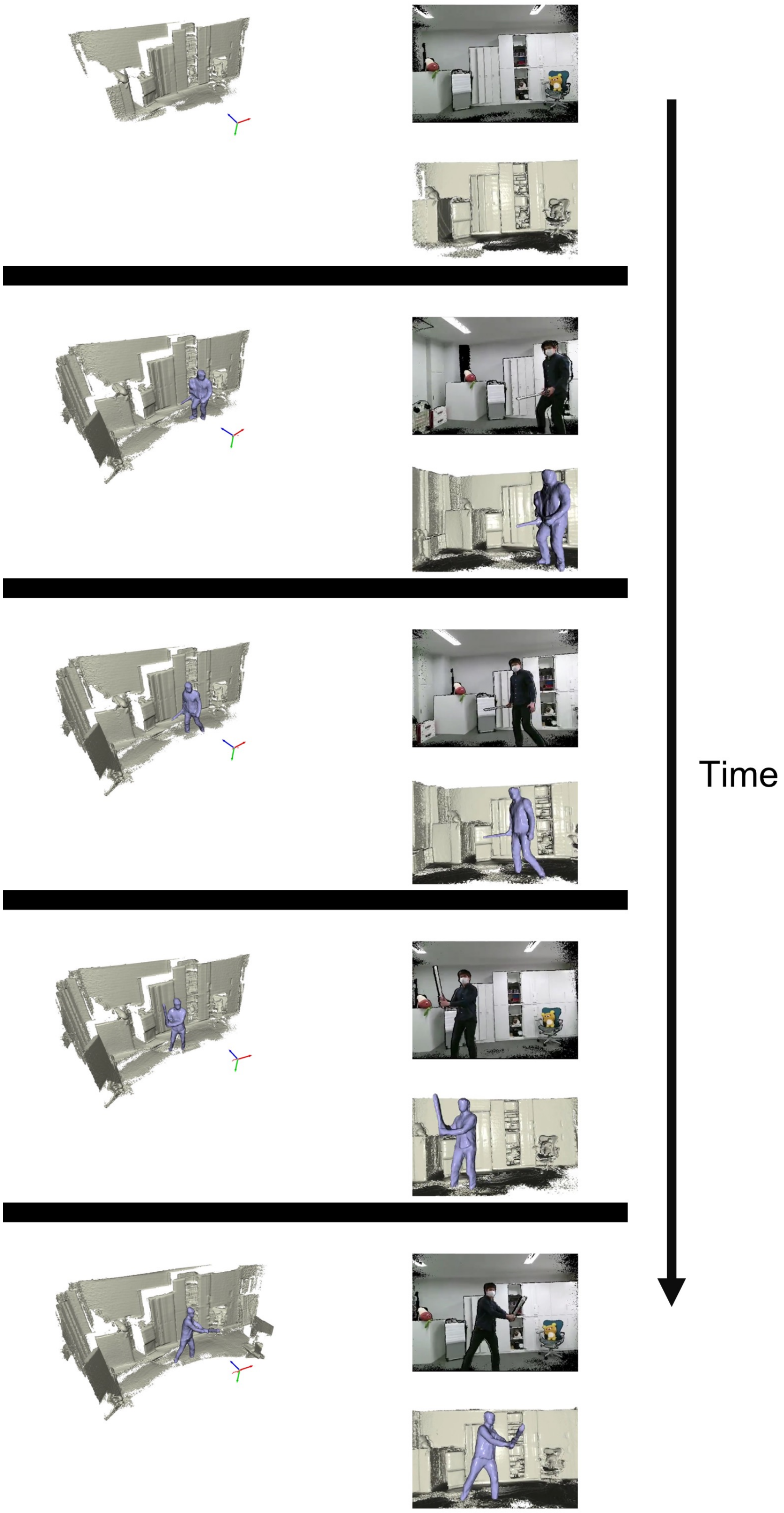


Figure 8.3 4D reconstruction of our lab. Left: the reconstructed scene; right-top: the input color image; right-bottom: the reconstructed scene that is projected to the current frame.

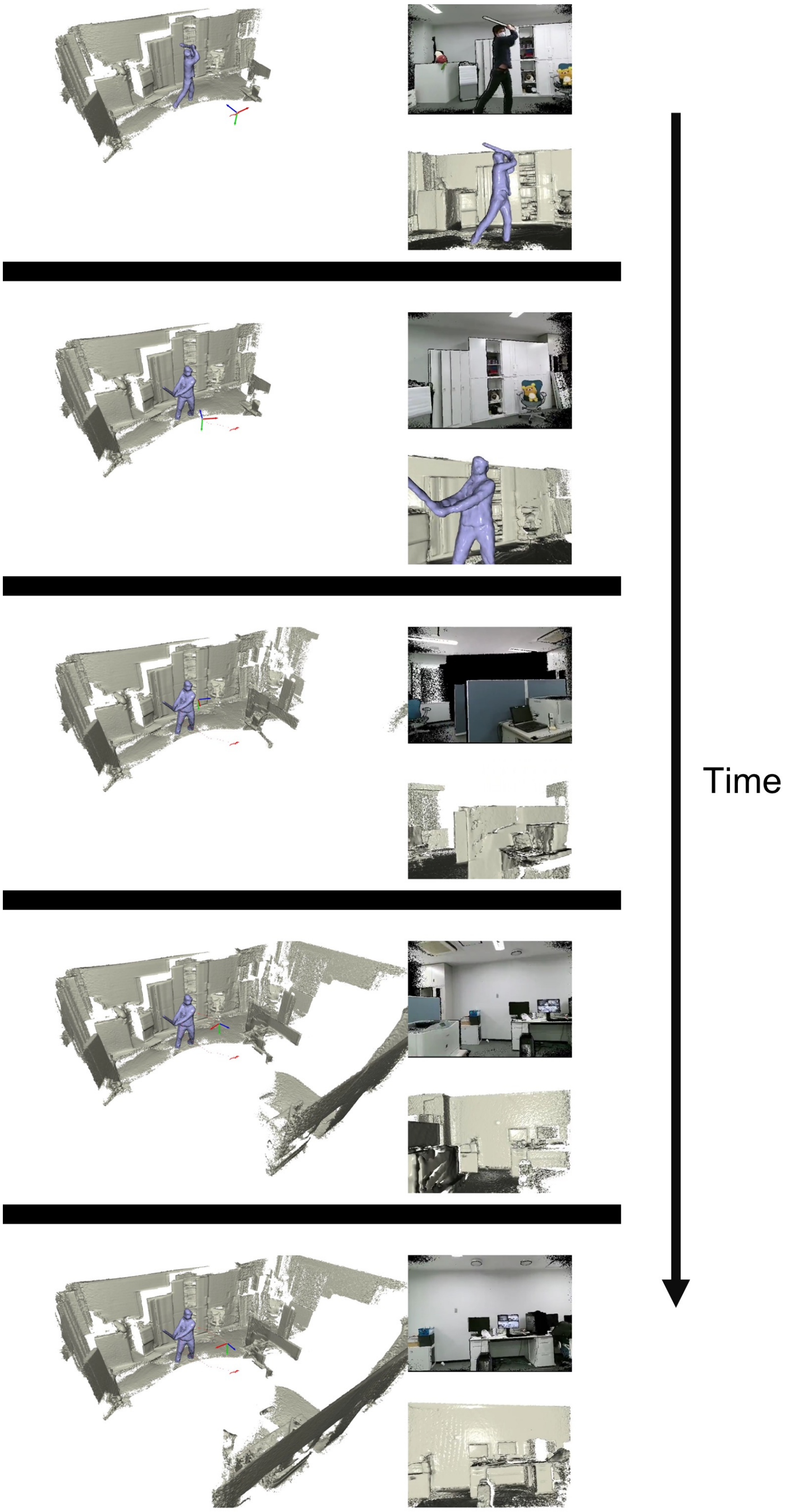


Figure 8.4 4D reconstruction of our lab. Left: the reconstructed scene; right-top: the input color image; right-bottom: the reconstructed scene that is projected to the current frame.

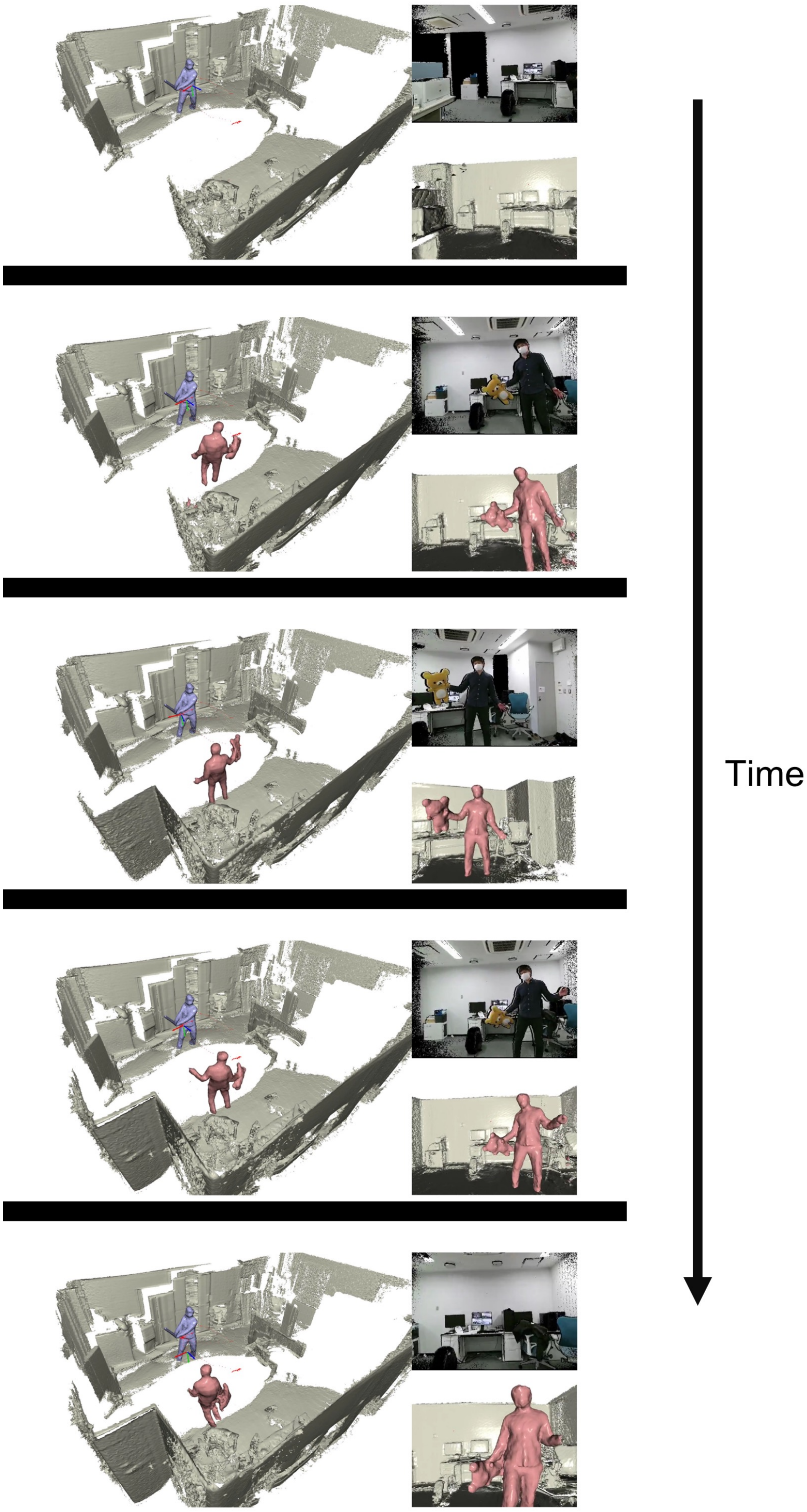


Figure 8.5 4D reconstruction of our lab. Left: the reconstructed scene; right-top: the input color image; right-bottom: the reconstructed scene that is projected to the current frame.

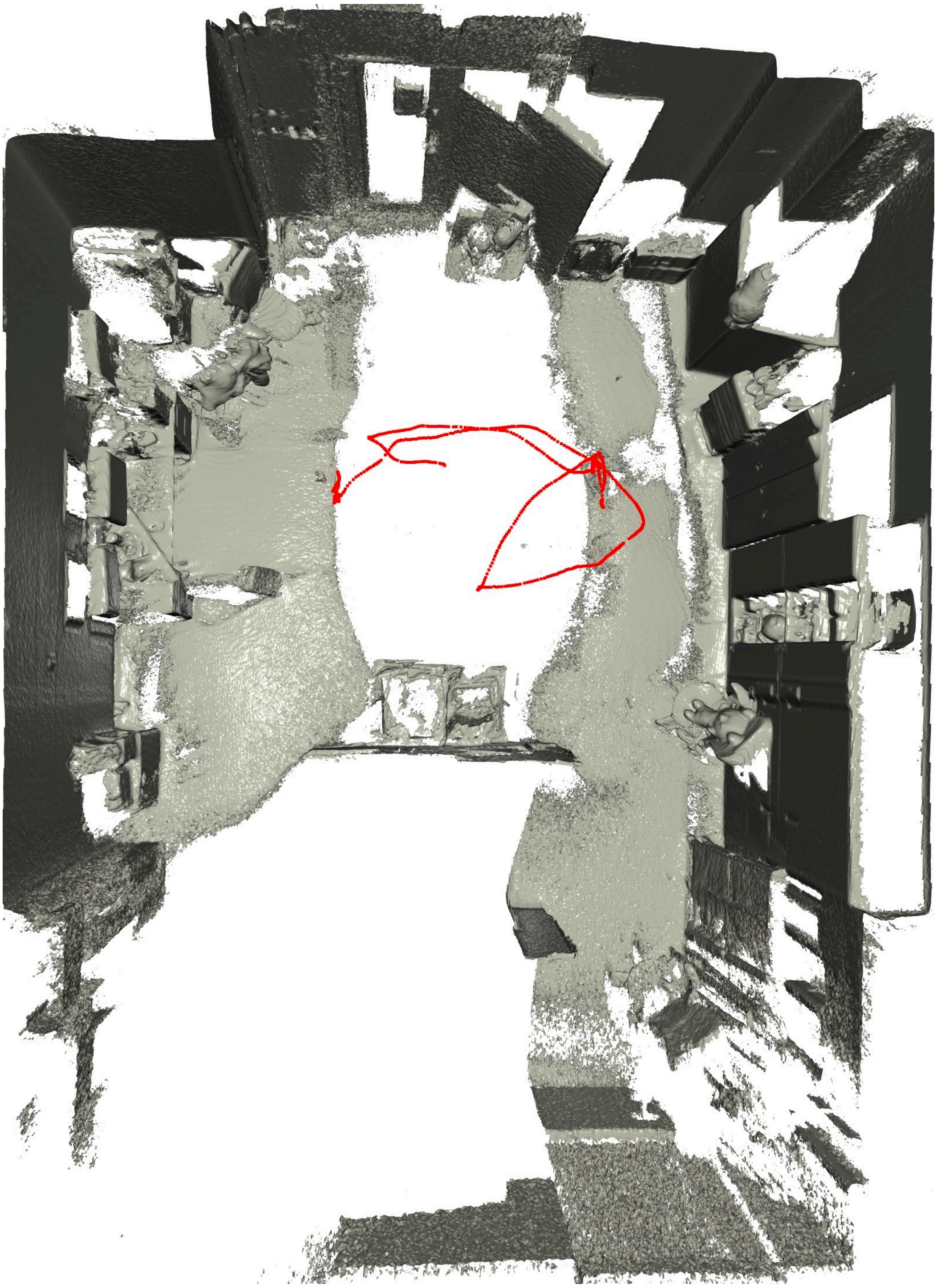


Figure 8.6 Top view of the reconstructed room. The red dotted line shows the trajectory of the Kinect2 camera.

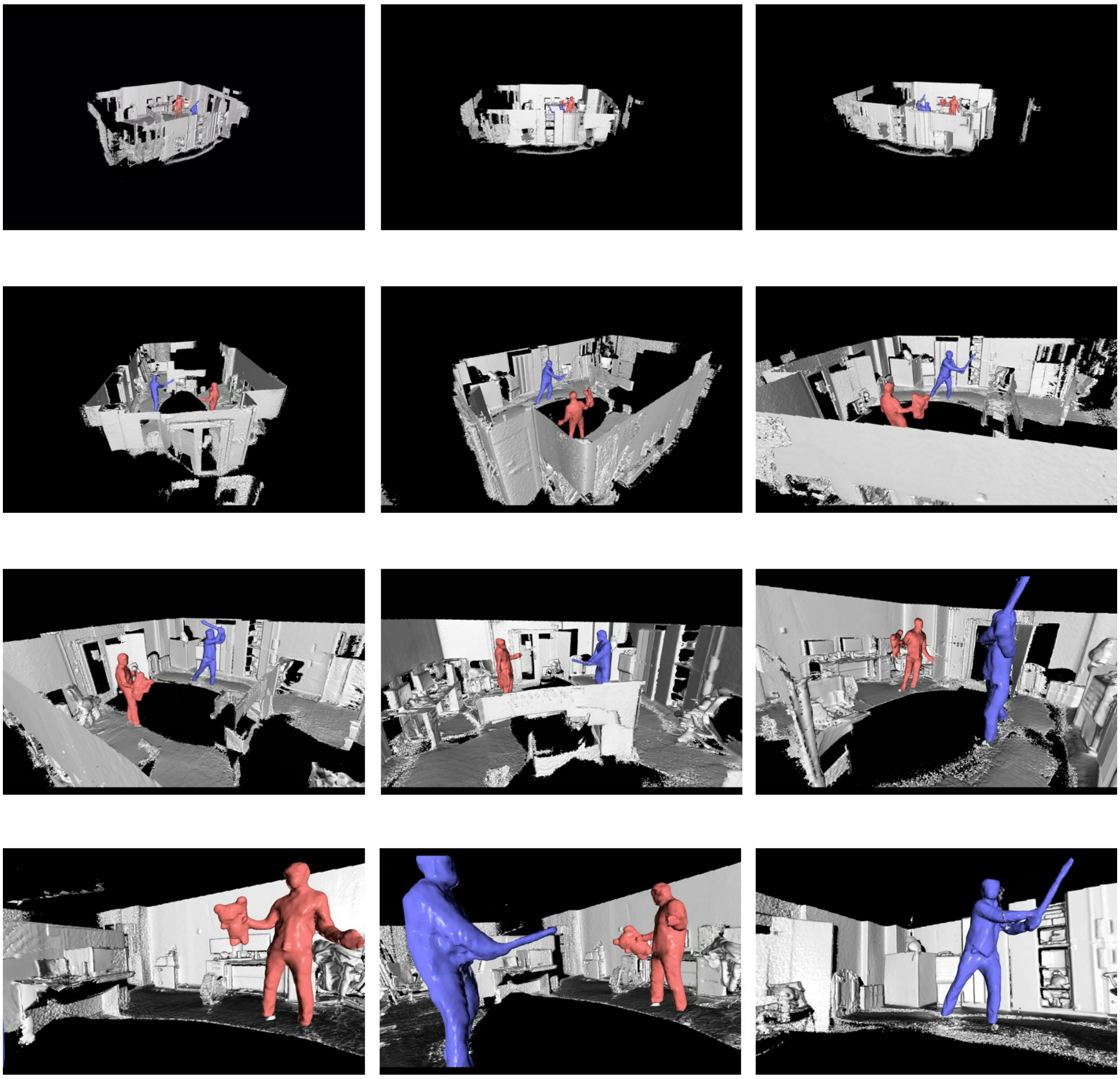


Figure 8.7 Flying through the 4D reconstruction of the lab. The two people are manually aligned in the time axis.

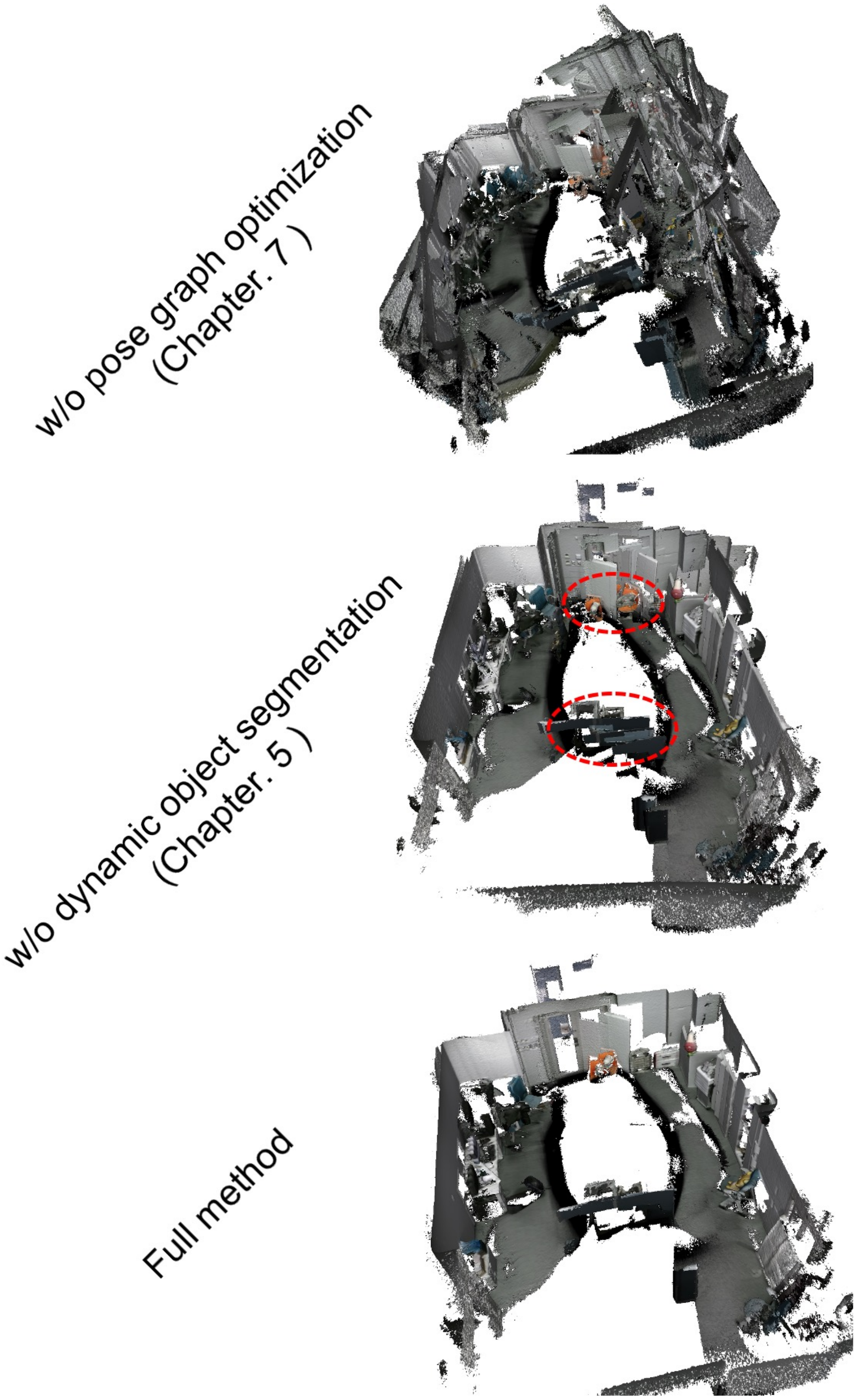


Figure 8.8 Ablation study for background reconstruction. Top: no pose graph optimization (c.f. Chapter 7). Middle: no dynamic object segmentation (c.f. Chapter 5). Bottom: our full method.

Chapter 9

Conclusion & Future Work

Non-rigid 4D reconstruction using a single RGB-D camera has numerous applications in computer vision, virtual/augmented reality, and robotics, etc. However, is it a very challenging problem because the problem itself is highly complex, and involves non-convex solving. In addition, a single RGB-D camera suffers from occlusion. In this thesis, we build a novel non-rigid reconstruction system to overcome certain limitations of this problem. This section summarizes our approaches and the knowledge we obtained through developing this system.

In chapter. 5, we proposed a framework call SplitFusion that reconstructs both static and dynamic scene elements by leveraging semantic instance segmentation. Experiments show that eliminating the dynamic foreground objects helps camera 6dof pose tracking in the non-rigid scenes. Moreover, reconstruction quality for both rigid and non-rigid element is significantly improved over the state-of-the-art SLAM method and non-rigid reconstruction method. We also found that semantic instance information informs the proposed SplitFusion system to handle the object-level topological change of the scene.

The core of non-rigid reconstruction is the non-rigid tracking problem. To develop robust tracking method that can handle large non-rigid motion, in chapter. 4, we present an end-to-end learning approach for non-rigid RGB-D tracking. We redefine the objective of non-rigid tracking by a deep feature fitting term which is learned end-to-end through a CNN network from a large amount of non-rigid dataset. Since the CNN kernels with a large receptive field can capture the global deformation, the learned deep feature fitting term significantly improves the convergence of non-rigid Gauss-Newton solver. The result tracking method can handle larger non-rigid motion than the classic method such as DynamicFusion, which employs a point-to-plane non-rigid ICP tracker. While we apply this method for non-rigid tracking, we believe the idea can be adapted to other tasks where it needs to solve least square problems, such as optical flow estimation.

The fundamental limitation of single RGB-D camera based reconstruction is the occlusion problem. A single depth camera can only measure the observable part of a given scene and usually cause surface disconnections and incompleteness. Occlusion also poses a challenge for motion tracking since the occluded geometry usually contains valuable information for tracking. To tackle this problem, in Chapter. 6, we introduce the first method that jointly recovers the high-resolution structure and motion field from partial observation. Through ablation studies, we found that understanding complete object geometry is beneficial for non-rigid tracking, and learning the motion of a scene is also beneficial for geometry completion. The proposed method also demonstrates better performance in estimating the occluded motion.

We believe that this thesis is a significant step toward building a robust non-rigid 4D reconstruction system that is available in people's daily life. We have shown the advantage of leveraging data-driven priors for 4D reconstruction in scene segmentation, robust tracking, and motion and geometry completion. Here, we list a few future directions:

- In Chapter. 4, we have developed a robust non-rigid tracking method to handle large deformation of the scene. While robust non-rigid tracking between frames with limited overlap area is yet a challenging problem and thus interesting. A potential solution is to learn the overlapping area between two frames while performs tracking.

- Dynamic/static scene decomposition is another interesting yet important direction. What could be interesting is to learn the decomposition in a self-supervised fashion by reasoning motion and scene contrast.

- In Chapter. 6, we have shown that our shape and motion completion method could recover reasonable structure from a single RGB-D frame. However, single view based completion method inevitably loses details for the unseen, for which it basically generates a averaged smooth geometry. The promising solution is to extend this method to a multi-view setup, where you can leverage multiview geometry information to achieve detailed completion.

- This direction is more ambitious: can we differentiate through the entire non-rigid reconstruction process and learn it end-to-end through data? This approach can leverage the newly constructed DeformingThings4D dataset (cf. Chapter 6), which we show is large enough to generalize to real-world sequences. This direction has the potential to revolutionize the field of generic non-rigid reconstruction.

References

[1] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007.

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[3] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 617–632, 2016.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features. *Computer vision and image understanding*, 110(3):346–359, 2008.

[5] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[6] Åke Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[7] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE International conference on computer vision (ICCV)*, 2019.

[8] Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural non-rigid tracking. *arXiv preprint arXiv:2006.13240*, 2020.

[9] Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7002–7012, 2020.

[10] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*, volume 2, pages 690–696, 2000.

[11] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[12] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 611–625, 2012.

[13] Jacobi Carl Gustav Jacob. Uber eine neue aufi¨osungsart der bei der methode der kleinsten quadrate vorkommenden linearen gleichungen. In *Nachrichten*, pages 22, 297, 1845.

[14] Che-Han Chang, Chun-Nan Chou, and Edward Y Chang. Clkn: Cascaded lucas-kanade networks for image alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2213–2221, 2017.

[15] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6970–6981, 2020.

[16] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019.

[17] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[18] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017.

[19] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(3):24, 2017.

[20] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5868–5877, 2017.

[21] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2018.

[22] Angela Dai, Christian Diller, and Matthias Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 849–858, 2020.

[23] Yuchao Dai, Hongdong Li, and Mingyi He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 107 (2):101–122, 2014.

[24] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[25] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016.

[26] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: Real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6): 1–16, 2017.

[27] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 834–849, 2014.

[28] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

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

[30] Wei Gao and Russ Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. *arXiv preprint arXiv:1904.13073*, 2019.

[31] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

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

[33] Aleksey Golovinskiy and Thomas Funkhouser. Min-cut based segmentation of point clouds. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 39–46. IEEE, 2009.

[34] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[35] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9224–9232, 2018.

[36] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[37] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3263, 2019.

[38] Oshri Halimi, Ido Imanuel, Or Litany, Giovanni Trappolini, Emanuele Rodolà, Leonidas Guibas, and Ron Kimmel. The whole is greater than the sum of its nonrigid parts. *arXiv preprint arXiv:2001.09650*, 2020.

[39] Lei Han, Mengqi Ji, Lu Fang, and Matthias Nießner. Regnet: Learning the optimization of direct image-to-image pose registration. *arXiv preprint arXiv:1812.10212*, 2018.

[40] Takeo Igarashi, Tomer Moscovich, and John F Hughes. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)*, 24(3):1134–1141, 2005.

[41] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 362–379, 2016.

[42] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[43] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[44] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6001–6010, 2020.

[45] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[46] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision*, pages 1–8. IEEE, 2013.

[47] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015.

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

[49] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[50] Suryansh Kumar. Jumping manifolds: Geometry aware dense non-rigid structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5346–5355, 2019.

[51] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Computer graphics forum*, volume 27, pages 1421–1430. Wiley Online Library, 2008.

[52] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7286–7291. IEEE, 2018.

[53] Yang Li, Aljaz Bozic, Tianwei Zhang, Yanli Ji, Tatsuya Harada, and Matthias Nießner. Learning to optimize non-rigid tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4910–4918, 2020.

[54] Zhe Li, Tao Yu, Chuanyu Pan, Zerong Zheng, and Yebin Liu. Robust 3d self-portraits in seconds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1344–1353, 2020.

[55] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019.

[56] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9246–9255, 2019.

[57] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[58] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

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

[60] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James M Rehg, and Jan Kautz. Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 468–484, 2018.

[61] Zhaoyang Lv, Frank Dellaert, James M Rehg, and Andreas Geiger. Taking a deeper look at the inverse compositional algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4581–4590, 2019.

[62] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

[63] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019.

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

[65] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5): 1147–1163, 2015.

[66] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011.

[67] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015.

[68] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5379–5389, 2019.

[69] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013.

[70] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019.

[71] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and P-P Sloan. Interactive ray tracing for isosurface rendering. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 233–238. IEEE, 1998.

[72] Mark Pauly, Niloy J Mitra, Joachim Giesen, Markus H Gross, and Leonidas J Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing*, pages 23–32, 2005.

[73] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *arXiv preprint arXiv:2003.04618*, 2020.

[74] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000.

[75] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[76] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571. Ieee, 2011.

[77] Martin Rünz and Lourdes Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4471–4478, May 2017.

[78] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018.

[79] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2016.

[80] Raluca Scona, Mariano Jaimez, Yvan R Petillot, Maurice Fallon, and Daniel Cremers. Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[81] Yifei Shi, Kai Xu, Matthias Nießner, Szymon Rusinkiewicz, and Thomas Funkhouser. Planematch: Patch coplanarity prediction for robust rgb-d reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 750–766, 2018.

[82] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661*, 2020.

[83] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1386–1395, 2017.

[84] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1754, 2017.

[85] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, volume 4, pages 109–116, 2007.

[86] Gilbert W Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, 17(3): 403–409, 1980.

[87] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE, 2012.

[88] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Transactions on Graphics (TOG)*, 26(3):80, 2007.

[89] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018.

[90] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.

[91] Osamu Tatebe. The multigrid preconditioned conjugate gradient method. 1993.

[92] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020.

[93] Henk A van der Vorst. High performance preconditioning. *SIAM Journal on Scientific and Statistical Computing*, 10(6):1174–1185, 1989.

[94] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics (TOG)*, 28(2):1–15, 2009.

[95] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[96] Shenlong Wang, Sean Ryan Fanello, Christoph Rhemann, Shahram Izadi, and Pushmeet Kohli. The global patch collider. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 127–135, 2016.

[97] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 91–98, 2020.

[98] Sebastian Weiss, Robert Maier, Daniel Cremers, Rudiger Westermann, and Nils Thuerey. Correspondence-free material reconstruction using sparse surface constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4686–4695, 2020.

[99] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. volume 35, pages 1697–1716. SAGE Publications Sage UK: London, England, 2016.

[100] Wenxuan Wu, Zhiyuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. *arXiv preprint arXiv:1911.12408*, 2019.

[101] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 817–833, 2018.

[102] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 467–483, 2016.

[103] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174. IEEE, 2018.

[104] Tao Yu, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7287–7296, 2018.

[105] Tao Yu, Zerong Zheng, Yuan Zhong, Jianhui Zhao, Qionghai Dai, Gerard Pons-Moll, and Yebin Liu. Simulcap: Single-view human performance capture with cloth simulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5499–5509, 2019.

[106] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1802–1811, 2017.

[107] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 340–349, 2018.

[108] Tianwei Zhang and Yoshihiko Nakamura. Posefusion: Dense rgb-d slam in dynamic human environments. In *International Symposium on Experimental Robotics*, pages 772–780. Springer, 2018.

[109] Tianwei Zhang and Yoshihiko Nakamura. Hrpslam: A benchmark for rgb-d dynamic slam and humanoid vision. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 110–116. IEEE, 2019.

[110] Tianwei Zhang, Huayan Zhang, Yoshihiko Nakamura, and Lei Zhang. Flowfusion: Dynamic dense rgb-d slam based on optical flow. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[111] Bo Zheng, Yibiao Zhao, Joey C Yu, Katsushi Ikeuchi, and Song-Chun Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3127–3134, 2013.

[112] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018.

[113] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1851–1858, 2017.

[114] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (ToG)*, 33(4):156, 2014.