Surfel-based RGB-D Reconstruction and SLAM with Global and Local Consistency

Yi (Jack) Yang

CMU-RI-TR-19-45

July 2019

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Michael Kaess (Chair)
Simon Lucey
Kumar Shaurya Shankar

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

For my family

Abstract

Achieving high surface reconstruction accuracy in dense mapping has been a desirable target for both robotics and vision communities. In the robotics literature, simultaneous localization and mapping (SLAM) systems use depth-enabled cameras to reconstruct a dense map of the environment. They leverage the depth input to provide accurate local pose estimation and a locally consistent model. However, drift in the pose tracking over time leads to misalignments and artifacts. On the other hand, offline computer vision methods, such as the pipeline that combines structure-from-motion (SfM) and multi-view stereo (MVS), estimate the camera poses by performing batch optimization. These methods achieve global consistency, but suffer from heavy computation loads.

We propose two novel approaches that integrate both methods to achieve locally and globally consistent reconstruction. The first method estimates the poses of *keyframes* in the offline SfM pipeline to provide strong global constraints at relatively low cost. Afterwards, we compute odometry between *frames* driven by off-the-shelf SLAM systems with high local accuracy. We fuse the two pose estimations using factor graph optimization to generate accurate camera poses for dense reconstruction.

The second method applies bundle adjustment to improve the estimation of both camera tracking and landmarks, while simultaneously optimizing the dense model upon loop closure using a deformation graph. Through efficient implementation on GPU, the system is able to achieve online performance and accurate dense reconstruction.

Experiments on real-world and synthetic datasets demonstrate that our approaches produce more accurate models comparing to existing dense SLAM systems, while achieving significant speedup with respect to state-of-the-art SfM-MVS pipelines.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Professor Michael Kaess. Without his unwavering support, patience, and guidance along the way, I would not be able to finish what I have done today. I would also like to thank my thesis committee, Professor Simon Lucey and Kumar Shaurya Shankar, for their help and suggestions.

Being a part of the Robot Perception Lab, I am grateful for the discussions, encouragement, and friendships. You are not only a source of inspiration, but also a beam of light in the sunless basement of NSH 2nd floor.

For many of my friends at CMU and Mudd, I want to let you know that I am thankful for all the help over the past two years. I would also like to give my special thank to Lizzi Yin for bringing me various of encouragements and laughter. Thank you all for being there.

Finally, I want to let my parents know that I would not be able to make it without your faith in me. Your unfaltering love and constant support empower me to overcome the hardships in my journey of life.

Contents

Chapter 1

Introduction

1.1 Motivation

Knowing the environment has been a crucial component in rising technologies such as mobile robotics, virtual reality, and various applications in artificial intelligence. For example, a construction robot requires a detailed map of the construction site to be able to navigate around the environment; infrastructure inspection requires recovering a realistic 3D model of the object; virtual reality needs to create vivid 3D models to provide an immersive experience to the users. The essence of all of these application is the task of creating a 3D model with accurate geometry to facilitate the perception of the environment.

The task of mapping and reconstruction has been addressed in various ways in the past. Without any prior knowledge of the robot state, mapping of the environment usually requires that we know where the sensor or camera is. This problem is called simultaneous localization and mapping (SLAM). If we cannot accurately detect the states of the sensor, we cannot trust the resulting map. Meanwhile, if we do not have a reliable map, estimating the state of the sensor becomes an intractable task. Therefore, the mapping problem is tightly coupled with the state estimation of the sensor, and it is important that both aspects are addressed at the same time.

There are many different representations of the map. For example, sparse point maps are used to encoded important information of the environment, such as landmarks, and provide state estimation information; topological maps such as Google navigation map are popular such that it provides navigation instructions. However, in autonomous robot applications and virtual reality, a dense 3D map is more desirable because it not only provides the sparse landmarks, but it also contains the information that can help a robot to navigate around and avoid the obstacles. In addition, infrastructure inspection using autonomous robots requires the sensors to recover an as-realistic-as-possible map of the environment; sparse points simply cannot be used to encode such information.

Meanwhile, as RGB-D and stereo cameras have become widely available, per-pixel level depth estimation has been greatly simplified, which is valuable for dense scene reconstruction. The

abundance of depth information has allowed the development of various algorithms and systems such as KinectFusion [1], VoxelHashing [2], Kintinuous [3], ElasticFusion [4], and Open3D [5]. In this thesis, we proposed an offline and an online reconstruction system that combine traditional 3D reconstruction methods in computer vision, dense SLAM, and computer graphics to generate high-fidelity 3D models with global and local consistency.

1.2 Current Challenges

The two key demands for a reconstruction or mapping algorithm to fulfill are:

- accurate modelling of the global geometry of a large scale environment in terms of global geometric consistency, and
- detailed scene geometry such as locally consistent shape, texture, and color.

Many efforts have been devoted to address both requirements [6], [7], [1], [3] [4]. These efforts can be divided into two categories: offline methods such as structure-from-motion (SfM) with multi-view stereo (MVS), and online methods such as dense 3D SLAM. Depending on the applications and requirements in efficiency, robustness, and accuracy, one of the methods can be employed. However, it is not uncommon that these methods fail in one of the criteria listed above.

Specifically, in computer vision methods such as the widely used SfM-MVS pipeline, the initial structure and camera poses are computed using feature points. Later, bundle adjustment (BA) is applied to optimize the overall structure and poses by minimizing the reprojection error. One of the challenges in SfM is the presence of outliers in BA. In reality, the outliers from either the mismatched features or the incorrect pose initialization produce large error. The process of BA distributes this error across all structures and poses, and results in inaccurate camera pose and structure estimations. In addition, the reconstruction systems sometimes reject the image input in a featureless region and resulting in inaccurate camera pose estimation for the corresponding images.

On the other hand, systems that are dedicated to generating detailed local geometry and accurate short-term odometry in a room-size scene, such as ElasticFusion [4], often fail at large scale operations due to tracking failures. Other offline or online dense RGB-D reconstruction systems such as [8] and [9] produce accurate models, but often at the cost of significant computation resources and time.

1.3 Contribution

This thesis presents two major contributions:

- We introduce a framework to combine SfM and SLAM in a factor graph formulation. Although our method requires offline pose optimization, it produces more accurate model geometry comparing to online methods, and is much less time-consuming comparing to other offline reconstruction systems.
- We introduce an online system that leverages the sparse landmarks to improve the dense map. To the best of our knowledge, this is the first work that links the optimization of the camera poses and the dense 3D model using surface deformation. In this thesis, we detailed the steps and algorithms to establish the formulation. In addition, we will make the code available to the public at `https://bitbucket.org/rpl_cmu/structurefusion/src/`.

The thesis is organized as three major components: an overview of the state-of-the-art dense mapping and SLAM systems, an offline reconstruction system that achieves state-of-the-art geometry accuracy in dense reconstruction, and an online system that maintains the global consistency for both the camera poses and the reconstructed model through elastic deformation.

Chapter 2

Related Work

3D reconstruction from images has been a historically well-explored topic. A plethora of methods and systems have been proposed over the past 30 years. Meanwhile, SLAM using visual sensors has gradually become one of the most popular method to explore unknown 3D environments. The mapping process, which involves constructing a 3D model of the unknown environment, has naturally overlapped with the topics in 3D reconstruction. The problem has gathered great attention from both the robotics and the vision communities.

In general, we can divide the 3D mapping and reconstruction into three different categories. First, structure-from-motion and multi-view stereo (MVS) has been thoroughly studied by the computer vision community. As the RGB-D camera has become widely available, dense offline RGB-D reconstruction using depth map has also been proposed. Finally, the real-time dense SLAM formulation has also become possible as parallel computation units such as GPU become widely available. The chapter provides an overview of all three categories of dense 3D reconstruction, as well as sparse landmark-based SLAM.

2.1 Structure-from-Motion

Undoubtedly, SfM-MVS pipeline has been one of the most popular method to recover 3D structure from 2D images. From early systems such as Bundler [10, 11] to later large scale 3D reconstructions using internet images [12], many systems address various problems from fundamental multi-view geometry to robust correspondent verification. Recently, systems such as PMVS [13], OpenMVG [6, 14], Theia [15], COLMAP [7, 16], OpenMVS [17] provides not only the multi-view reconstruction algorithms but also a rich user interface that allow users to interact with the reconstruction process. These systems address various problems and concerns in the early stage of computer vision, and greatly improve the robustness, completeness, accuracy, and efficiency.

The problem of accuracy and efficiency has been addressed in many large scale offline SfM-MVS pipelines and online dense mapping systems. One of the most popular approaches is based on the

idea of divide-and-conquer. Zhu et al. [18] and Yao et al. [19] introduce the technique of locally readjusting the camera poses to improve the reconstruction quality. This method is effective in terms of smoothing out the rough surfaces and recovering more accurate local geometry. However, these methods still suffer from predefined heuristics such as the size of local camera group. Built upon these ideas, many state-of-the-art SfM pipelines such as COLMAP by Schönberger et al. [16] and OpenMVG by Moulon et al. [6] have developed complete systems that apply local refinement to recover better local reconstruction quality. These state-of-the-art systems provide globally accurate camera pose estimation.

While many systems intend to solve the problem of SfM, using purely 2D images has some drawbacks. First, it heavily relies on 2D features to establish correspondence. Features can be brittle in the context of repetitive textures, textureless scenes, and varying lighting conditions. Second, multiple bundle adjustments and geometric verification form the core of those SfM methods, which is often time consuming. Third, patch matching stereo methods and point cloud densification can be even more time consuming. The reconstructed results often have "holes", e.g. missing areas, on textureless surfaces due to a lack of features. Finally and most importantly, these methods sometimes are prone to outliers in the feature matching process. During the process of bundle adjustment, the error introduced by these outliers will be evenly distributed to all landmarks, and often results in local artifacts in the final reconstruction. We address this problem as the local consistency problem, and an example is provided in Fig. 2.1.

2.2 Dense Offline RGB-D Reconstruction

As RGB-D sensors became widely available over the years, many offline systems are developed to use the depth information to generate highly accurate 3D models. For example, Zhou and Koltun [20] propose to construct small fragments of a large scene, and then use volumetric registration to combine these small fragments. In addition, Zhou et al. [21] apply elastic regularization to merge all fragments. Apart from these methods, Choi et al. [8] apply a similar idea of constructing many overlapping small fragments, and then use Iterative Closest Point (ICP) to register the fragments together to complete a full scene. Recently, Open3D [5] introduces various algorithms to reconstruct the 3D environment with RGB-D information. However, these systems still face problems such as requiring manual alignment of the unregistered camera frames from separated fragments, and poor initialization of the camera poses could lead to incorrect convergence of the camera poses. These works produce accurate models, but at a cost of hours or even days of processing time.

2.3 Sparse Mapping in SLAM

Given the rich visual information from a camera, visual SLAM has been a popular field since the first single camera SLAM [22] system. As Klein et al. proposed the new PTAM [23] system that separate the mapping and tracking into two parallel threads, visual SLAM has become the

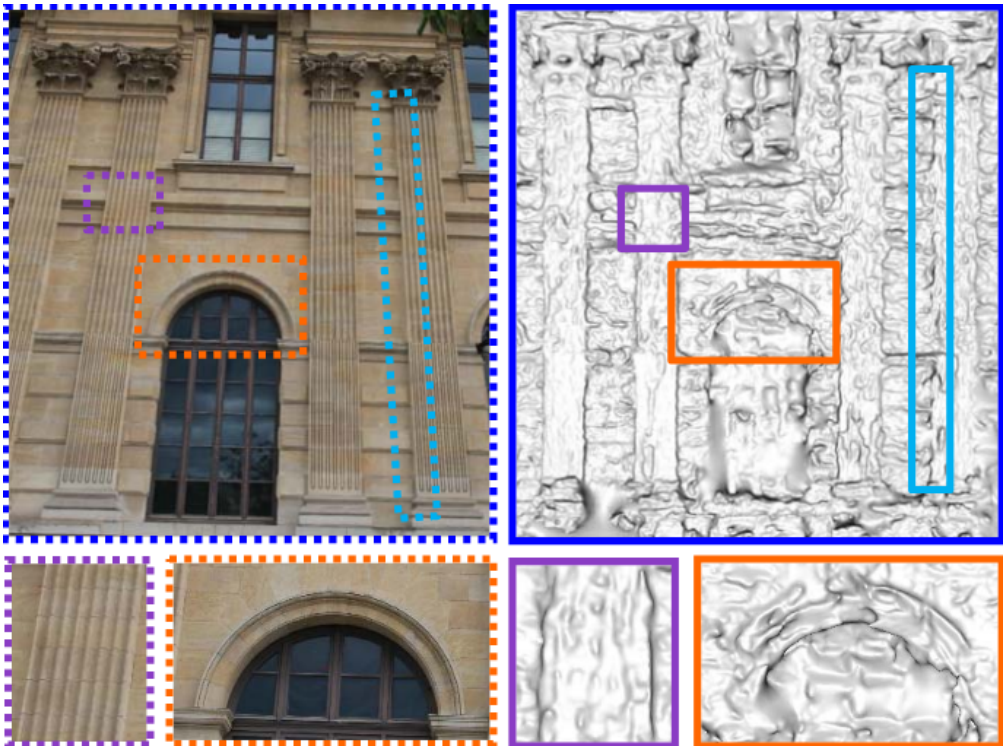


Figure 2.1: A problem of the local consistency shown by Zhu et al. [18]. The picture on the left shows the original 3D structure, and the picture on the right shows the reconstructed result from SfM-MVS pipeline. A lack of local consistency can be seen that the resulting surface is bumpy and irregular.

Figure 2.2: An example of failed global consistency of the 3D model reconstructed by ElasticFusion [4]. Although locally it preserves the model details, the model becomes distorted due to odometry drift and failed loop closure.

choice for robots to navigate in an unknown environment. Improving upon the work of PTAM, many visual SLAM systems such as ORB-SLAM [24] and LSD-SLAM [25] have been prevalent in real-world applications such as augmented reality, autonomous driving, and search and rescue. Although these SLAM systems provide relatively good camera tracking in real-time, the map that these systems build is represented by sparse points, which has less information than a dense map such as voxel grid or mesh. It is difficult to use the sparse map for robotics applications such as path planning and semantic understanding. Building upon these SLAM methods, dense map representations are needed in order for a robot to carry out meaningful actions.

2.4 Dense Online RGB-D Reconstruction

With the rise of RGB-D sensors such as Microsoft Kinect, Asus Xtion Pro, and Intel RealSense, dense SLAM with RGB-D sensor has become possible. As a subject of great interest, systems such as KinectFusion [1], Kintinuous [3], InifiniTAM [26, 27] focus on the surface reconstruction of the model using truncated signed distance function (TSDF) [28] that stores the signed distance from every voxel to its nearest surface. Although these systems are able to achieve high performance in a room-size reconstruction, they rely on the depth information to conduct frame-to-model ICP, which is brittle when the depth measurements are subject to significant sensor noise. As a result, they are vulnerable to accumulated drift. ElasticFusion [4] incorporated the color information in the frame-to-model tracking to minimize the effect of ICP drift. However, the ICP drift still exists when the scene is large. Another approach applied by the state-of-the-art BundleFusion [9] is the SIFT feature correspondence search and BA that is similar to sparse SLAM works such as ORB-

SLAM [24]. Although BundleFusion achieves superior performance in indoor reconstruction, it is computationally demanding, requiring two high-end GPUs to run in real-time. FlashFusion [29] applies a similar approach to use ORB [30] feature correspondence and keyframe integration without GPU. It achieves a similar performance to BundleFusion with much lower computation needs, but its keyframe integration strategy leads to a lower surface area coverage comparing to other dense mapping systems; it substantially sub-samples the frames and voxels. Most importantly, a common characteristic in all of the dense mapping methods in SLAM is that they often fail at a larger scene due to incorrect odometry estimation or incorrect loop closures. This results to inconsistent global model, as shown in Fig. 2.2.

Chapter 3

Preliminaries

3.1 Overview of Feature-based Visual Odometry and Structure-from-motion

Structure-from-motion (SfM) and feature-based visual odometry (VO) are the dominant methods to estimate the pose of the camera. The general pipeline for the feature-based methods can be listed as follows:

Feature Extraction: For each image, a feature extractor will extract features that represent the ”high gradient” region of the image. Each 2D feature on the image corresponds to a 3D object that is called ”landmark.” The feature extractor also generates a feature descriptor that is later used in feature matching. Common features include SIFT [31], SURF [32], and ORB [30].

Feature Matching: Between two images or between an image and a scene, a feature matcher finds and recognizes similar features and establishes the correspondence between these features based on their descriptors.

Geometric Verification: Since feature matching can only check the appearance-based feature similarity, it does not ensure that two corresponding features are exactly the same landmark. Therefore, it is necessary to verify the correspondence using geometric information. Usually, fundamental matrix or essential matrix is used to verify the geometric relationship between two sets of corresponding features. In addition, homography is also used to verified in pure rotation scenario. RANSAC [33] and its derivative methods are used in this stage to exclude any outlier correspondence.

Image Registration: Once two images are successfully matched and verified, we obtained a collection of landmarks that is called the map. Successively, we can register the image onto the map by applying feature matching and verification. Usually, algorithms such as Perspective-n-Point (PnP) [34] and its variants are used to register the image and obtain its corresponding camera pose.

Triangulation: Each image comes with some newly observed points that do not exist in the collection of landmarks. Once confirmed that there exist enough correspondences among images, a new point can be triangulated into the map.

Bundle Adjustment: During image registration and triangulation, it is unavoidable that some uncertainties are involved, and these accumulated uncertainties must be addressed in order to minimize the drift in the camera pose estimations. Here, bundle adjustment (BA) is used to minimize the reprojection error across all the landmarks by optimizing both the landmark positions and camera poses.

$$E_{BA} = \sum_{j} \sum_{i} \rho(\|\pi(P_i, \mathbf{X}_i) - \mathbf{x}_{ij}\|_{\Sigma}^2) \tag{3.1}$$

Here, ρ is a robust estimator that is used against the outliers, and $\pi(\cdot)$ is the projection function that projects a landmark onto an image. Common optimization methods such as Gauss-Newton and Levenberg-Marquardt are used in the optimization.

3.2 Joint Geometric and Photometric Tracking

3.2.1 Iterative Closest Point Tracking

Iterative Closest Point (ICP) is an algorithm of finding the transformation between two sets of 3D points. If the exact correspondence is given, the solution can be obtained by algebraic methods such as singular value decomposition in one step. However, in most cases, we are given two sets of 3D points with unknown correspondences. Therefore, the closest point pair is often used as the known correspondence. However, it is often inefficient to search for the closest neighbors using brute force. While data structures such as k-d tree [35] can be used to speed up closest neighbor searching, it contains many sequential instructions and requires random memory access, which is more difficult to implement on GPU. Here, we apply the *projective data association* to find the correspondence. Projective data association uses the normal vector of a point to project from the previous estimation to obtain the cross point, and uses the cross point as the corresponding point, as shown in Fig. 3.1.

Using projective data association, we minimize the point-to-plane distance defined as the following:

$$E_{icp} = \sum_{\mathbf{u}} \left\| (\mathbf{v}_k(\mathbf{u}) - \exp(\hat{\xi})\mathbf{T}\mathbf{v}_{k-1}(\mathbf{u})) \cdot \mathbf{n}_{k-1} \right\|^2, \tag{3.2}$$

where the exponential map $\exp(\hat{\xi})$ maps a member of the Lie algebra $\mathfrak{se}(3)$ to a member of Lie group SE(3). Here, $\mathbf{n}_{k-1}$ is the normal vector at time stamp $k-1$, and $\mathbf{v}_k$ is the vertex at time stamp k, $\mathbf{T}$ is the transformation from camera pose $\mathbf{T}_{k-1}$ to $\mathbf{T}_k$.

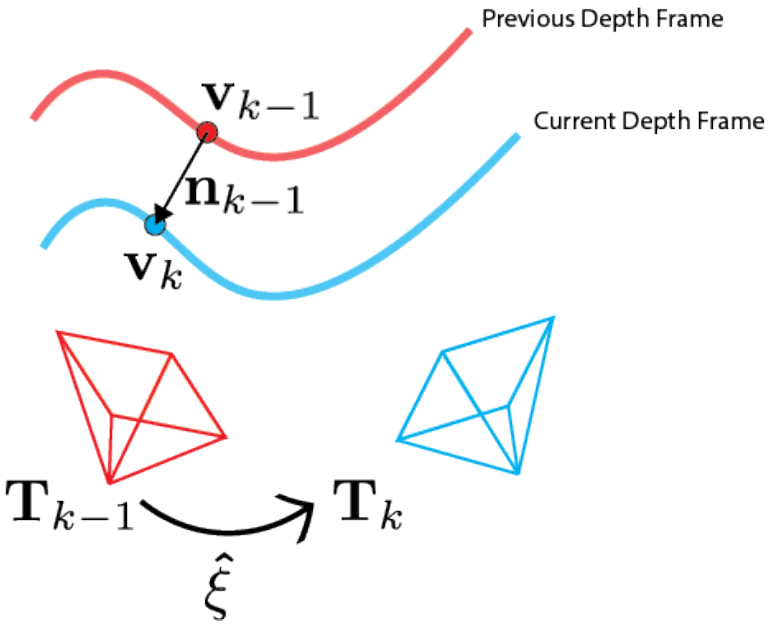


Figure 3.1: ICP projective data association. The red lines are the previous depth frame, and the blue lines are the current frame. Given a point $\mathbf{v}_{k-1}$, we can use the corresponding normal vector $\mathbf{n}_{k-1}$ to find the corresponding point along the direction of the ray.

3.2.2 Appearance-based Photometric Tracking

Given that the depth measurement from the sensor is noisy, it is not always reliable to use ICP as the sole method in estimating the camera pose. Therefore, we introduce the dense direct method based on the assumption that grey scale of two successive images are invariant. We propose the following formulation to calculate the pose transformation between two images by minimizing the photometric error:

$$E_{rgb} = \sum_{\mathbf{u}} \left\| I_t(\mathbf{u}) - I_{t-1}(\pi(K \exp(\hat{\xi}) \mathbf{T} \mathbf{p}(\mathbf{u}))) \right\|^2. \tag{3.3}$$

Here, the $\mathbf{p}(\mathbf{u})$ is the 3D point from the previous frame, and $\mathbf{T}$ is the pose transformation from previous frame to the current frame.

3.2.3 Joint Optimization of ICP and Photometric Tracking

We express the total error as a function of the joint ICP and photometric tracking term as below:

$$E_{track} = wE_{icp} + (1 - w)E_{rgb}, \tag{3.4}$$

where w is the relative weighting between the geometric and photometric term. To solve the system, we apply Gauss-Newton optimization to solve the system iteratively. We apply first order

linearization to the errors:

$$E_{rgb}(\hat{\xi} + \Delta\xi) \approx E_{icp}(\hat{\xi}) + J_{icp}(\hat{\xi})\Delta\xi, \tag{3.5}$$

$$E_{icp}(\hat{\xi} + \Delta\xi) \approx E_{rgb}(\hat{\xi}) + J_{rgb}(\hat{\xi})\Delta\xi, \tag{3.6}$$

where J_{icp} and J_{rgb} are the corresponding Jacobian matrices of the geometric and photometric term. The problem is reduced to solving a least-square system represented as:

$$\sum_u \mathbf{A}\Delta\hat{\xi} = -\sum_u \mathbf{b}, \tag{3.7}$$

$$\hat{\xi} = \Delta\xi \oplus \hat{\xi}, \tag{3.8}$$

where the system is built up with

$$\mathbf{A} = (1-w)\ J_{rgb}^{\top} J_{rgb}(\hat{\xi}) + w\ J_{icp}^{\top} J_{icp}(\hat{\xi}), \tag{3.9}$$

$$\mathbf{b} = (1-w)\ J_{rgb}^{\top} E_{rgb}(\hat{\xi}) + w\ J_{icp}^{\top} E_{icp}(\hat{\xi}). \tag{3.10}$$

A coarse-to-fine image pyramid [36] is implemented to accelerate convergence.

We implement the tracking in GPU given that the problem can be divided into pixels. Each pixel contributes the same to the entire Jacobian. We implement an efficient block sum reduction algorithm to reduce the $J^{\top}J$ array into a single 6×6 matrix and solve for the final result by Cholesky decomposition on the CPU.

Chapter 4

Surfel-Based Dense RGB-D Reconstruction with Global and Local Consistency

In Chapter 2, we introduce some of the most prevalent reconstruction systems used by both the computer vision and robotics community. However, these reconstruction works often fail to address some of the existing problems. For example, in SfM, structure estimation from multi-view geometry is usually prone to outliers, and procedures such as patch matching in MVS are time-consuming. The resulting mesh from SfM-MVS pipeline usually requires post-processing that is not only time-consuming but also labor intensive. On the other hand, dense SLAM systems often fail at large reconstruction due to camera pose drift. These failures often lead to misalignments in the final reconstruction.

In this chapter, we propose a new system that combines the advantages from both sides to achieve high fidelity surface reconstruction. We first solve a sparse problem of SfM by obtaining some of the camera *keyframe* poses, and then we use the frame-to-model odometry to recover the rest of the *frames'* poses. This method not only ensures that the camera does not drift due to odometry error accumulation, it also recovers the local details of the reconstruction that is often omitted in large scale SfM system.

4.1 Globally Consistent Pose Estimation using Factor Graph

Given a set of N RGB-D images captured by a consumer-grade RGB-D camera, keyframe camera pose priors are obtained through a SfM pipeline. Later, the camera poses are jointly optimized by the prior and odometry from the dense tracking. The optimized camera poses are used in the final reconstruction. Fig. 4.1 shows an illustration of the system architecture.

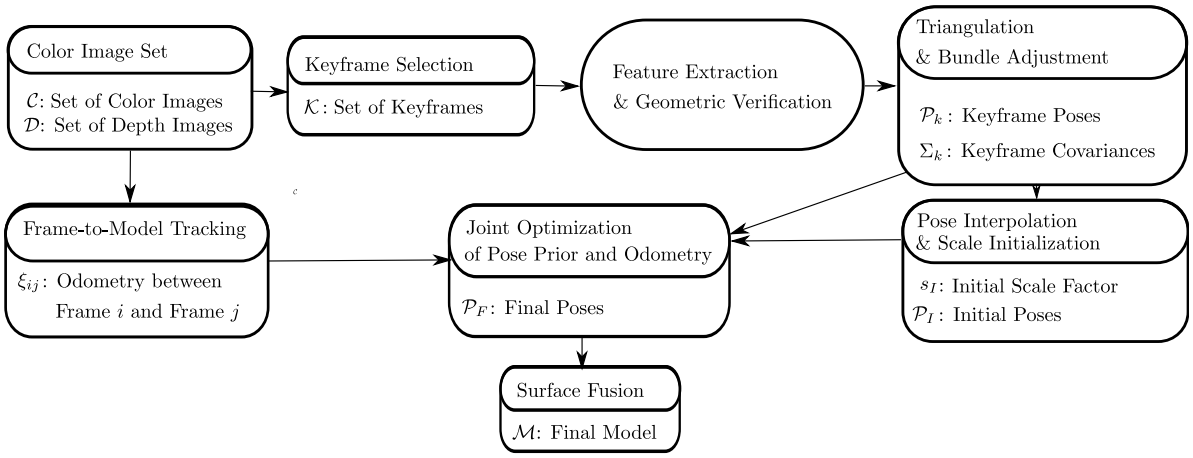


Figure 4.1: System block diagram. The top of each block shows the sub-components for each system, and the bottom of the block shows the output of the corresponding system. From a set of RGB-D images, a set of *keyframes* $\mathcal{K}$ is selected, and processed using the SfM pipeline. The resulting set of keyframe poses $\mathcal{P}_k$ and the corresponding covariance Σ_k are generated; these are treated as strong priors in the later optimization. Frame-to-model tracking provides odometry using joint optimization of ICP and photometric information. In preparation to the joint optimization of prior and odometry, keyframes are interpolated to provide a good initial value for the optimization. We optimize the poses incrementally as the new odometry becomes available, and fuse the new surfels from each frame into the existing model.

4.1.1 Prior and Odometry Estimation

Keyframe Pose Estimation: The set of color images $\mathcal{C}$ is evenly down-sampled with an interval of λ. The resulting color maps are regarded as *keyframes* $\mathcal{K} \subset \mathcal{C}$. Using the state-of-the-art system COLMAP [16], we estimate the i-th frame pose $\mathbf{P}_i \in \mathrm{SE}(3)$ and its corresponding covariance $\Sigma_{\mathbf{P}_i}$. The resulting camera pose is regarded as the camera pose prior in the optimization in Sec. 4.1.2. We refer to section 3.1 for a detailed overview of the Structure-from-Motion systems. We also refer to Schönberger et al. [7] and Moulon et al. [6] for a detail description of the systems such as COLMAP and OpenMVG.

Frame Odometry Estimation: The pose change between the i-th and j-th consecutive frames $\xi_{ij} \in \mathbb{R}^6$ is estimated using a joint frame-to-model [37] optimization. The cost function over the relative pose joins the point-to-plane distance and the photometric error, and is minimized in image pyramids for faster convergence. For details, we refer to section 3.2.

4.1.2 Factor Graph Representation

At the core of our proposed system is the joint optimization of the keyframe pose prior and frame-to-model odometry. To bring these together, we propose to use the factor graph formulation shown in Fig. 4.2. Specifically, given a keyframe $\mathbf{K}_i$ where $i \in \{1, 2, \cdots \lfloor \frac{N}{\lambda} \rfloor\}$ and its corresponding pose $\mathbf{P}_i$ estimated from SfM, we treat the keyframe pose as the pose prior in the factor graph. In addition, we represent the frame-to-model odometry ξ_{ij} between frame i and j as an odometry

factor. Since the keyframe prior and the frame-to-model odometry are independently estimated, these two kinds of factors are not aligned in the same coordinate frame. In addition, due to the scale ambiguity issue in SfM, we need to simultaneously estimate the scale that best aligns the prior and the odometry. The connections of the prior factor and the odometry factor are shown in Fig. 4.2. Given this formulation, the optimal solution of the problem is the *maximum a posteriori* estimation of the factor graph:

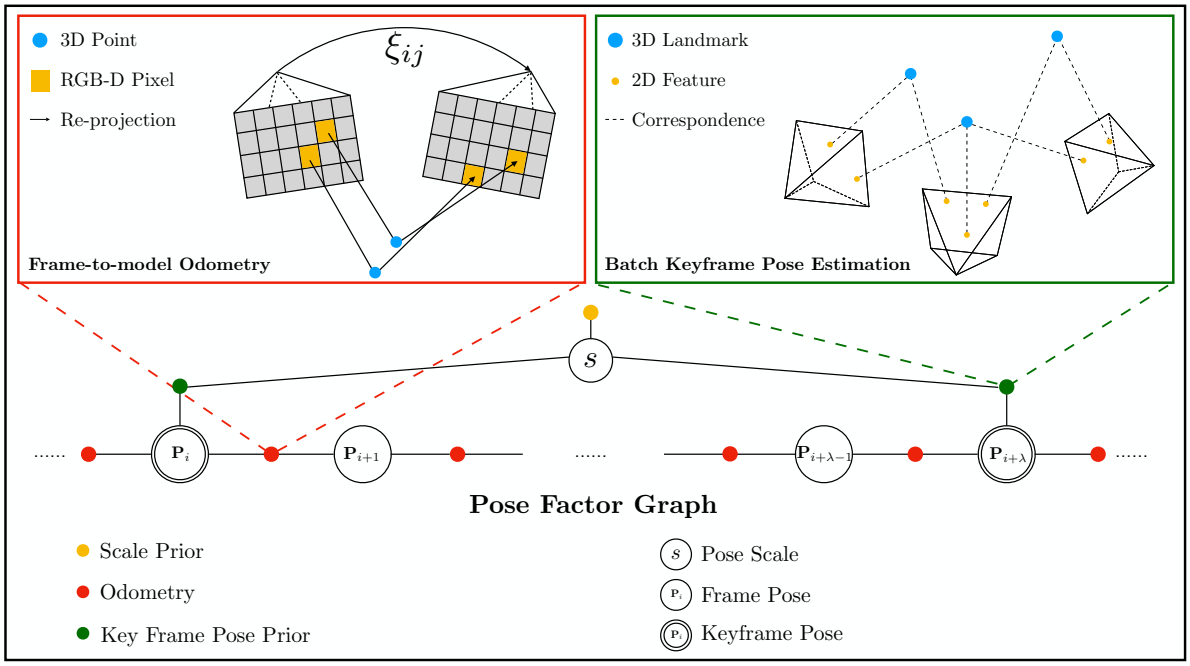


Figure 4.2: Frame-to-model odometry is solved using joint ICP and photometric optimization, as shown in the picture in the top left. The keyframe pose priors are solved using SfM, as shown in the top right. The factor graph in the bottom demonstrates how to combine the keyframe priors with the odometry.

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{i=1}^{N_K} \|r_s\|^2_{\sigma_s^2} + \sum_{i=1}^{N} \|\mathbf{r}_{\mathbf{P}_i}\|^2_{\Sigma_{\mathcal{P}_i}} + \sum_{i,j=1}^{N} \left\|\mathbf{r}_{\xi_{ij}}\right\|^2_{\Sigma_{\xi_{ij}}}, \tag{4.1}$$

where the set of optimal state is the set of all of the optimal poses and the optimal scale:

$$\mathcal{X}^* = \{\mathcal{P}^*, s^*\}. \tag{4.2}$$

r_s, $\mathbf{r}_{\mathbf{P}_i}$, and $\mathbf{r}_{\xi_{ij}}$ correspond to the residuals of scale, pose prior, and odometry respectively. σ_s^2, $\Sigma_{\mathbf{P}_i}$, and $\Sigma_{\xi_{ij}}$ correspond to the covariance of the scale, the pose prior, and the odometry. This problem can be solved using Levenberg-Marquardt non-linear optimization to iterate at the linearization point $\hat{\mathcal{X}}^k$. At the k-th iteration, the linearization can be expressed as the following Taylor expansion:

$$r_{s_i}(\hat{\mathcal{X}}^k + \delta\hat{\mathcal{X}}^{k+1}) \approx r_{s_i}(\hat{\mathcal{X}}^k) + H_{s_i}^k \delta\hat{\mathcal{X}}^{k+1}, \tag{4.3}$$

$$\mathbf{r}_{\xi_j}(\hat{\mathcal{X}}^k + \delta\hat{\mathcal{X}}^{k+1}) \approx \mathbf{r}_{\xi_j}(\hat{\mathcal{X}}^k) + H_{\xi_j}^k \delta\hat{\mathcal{X}}^{k+1}, \tag{4.4}$$

where we use the first order term to approximate the actual value. The measurement Jacobian H at the k-th iteration and the corresponding state update can be expressed as:

$$H_{s_i}^k = \frac{\delta r_{s_i}}{\delta \mathcal{X}}\bigg|_{\mathcal{X}=\hat{\mathcal{X}}^k}, \; H_{\xi_{ij}}^k = \frac{\delta \mathbf{r}_{\xi_{ij}}}{\delta \mathcal{X}}\bigg|_{\mathcal{X}=\hat{\mathcal{X}}^k}, \tag{4.5}$$

$$\hat{\mathcal{X}}^{k+1} = \hat{\mathcal{X}}^k \oplus \delta\mathcal{X}^{k+1}. \tag{4.6}$$

We use the $\oplus$ operator to represent retraction.

4.1.3 Pose Initialization: Interpolation on SE(3) Manifold

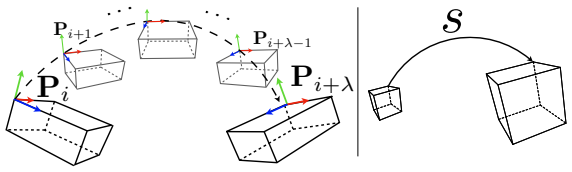


Figure 4.3: (Left) Illustration of the pose interpolation on manifold. (Right) 3D similarity transformation.

Initialization is of paramount importance to solve the non-convex optimization problem. It is important to initialize the state within close proximity of the optimal value to allow final convergence. If the sub-sampled interval λ is large, using the keyframe pose prior to initialize the intermediate poses is likely inaccurate, and thus might lead to wrong convergence. We propose to use an on-manifold interpolation based on the method shown in Žefran and Kumar [38] to initialize the poses in between two keyframes.

Let two keyframe poses be $\mathbf{P}_i$ and $\mathbf{P}_{i+\lambda}$ corresponding to the i-th and the $(i+\lambda)$-th frames, we want to find the intermediate pose $\mathbf{P}_k$ corresponding to the k-th frame that satisfies $i \leq k \leq i+\lambda$. This can be achieved by first finding the difference $\delta\mathbf{P}$ between two poses, and then map the difference from SE(3) to $\mathfrak{se}(3)$ represented by $\delta\xi$. Given the tangent space of the Lie manifold preserves linearity, we can estimate the intermediate poses corresponds to k as:

$$\mathbf{P}_k = \mathbf{P}_i \exp\left(\frac{k-i}{\lambda}\delta\xi\right). \tag{4.7}$$

4.1.4 Scale Initialization

Due to the scale ambiguity in the SfM, we need to find the scale factor $s \in \mathbb{R}$ that best aligns the two sets of poses. Similar to the pose initialization in Sec. 4.1.3, we also need to make sure that the scale is correctly initialized. We propose to use the 3D landmark $\mathbf{X}_{sfm}$ captured by the first frame in the SfM pipeline. Assuming that the first camera frame should be aligned with the global coordinate, we have the pose for the first frame expressed in the global coordinate frame to

be $\mathbf{P} = \mathbf{I}_{4\times4}$. The set of first frame landmarks $\mathbf{X}_{sfm}$ corresponds to the set of 2D pixel coordinates $\mathbf{x}_1$. We can extract the corresponding 3D landmarks $\mathbf{X}_{slam}$ from the first frame depth image using the following camera projection function:

$$\mathbf{X}_{slam} = \pi^{-1}(\mathbf{P}_1 = \mathbf{I}_{4\times4}, \mathbf{x}_1), \tag{4.8}$$

where $\pi^{-1}(\cdot,\cdot)$ is the inverse camera projection that maps the pixel coordinate to the world coordinate. Then, based on the method by Horn [39], we estimate a similarity transformation between two sets of 3D points using singular value decomposition:

$$\mathbf{U}, \mathbf{S}, \mathbf{V} = \text{svd}(\tilde{\mathbf{X}}_{sfm} \otimes \tilde{\mathbf{X}}_{slam}), \tag{4.9}$$

$$\mathbf{R} = \mathbf{U}\mathbf{V}^\top, \tag{4.10}$$

$$\tilde{\mathbf{X}}_{aligned} = \mathbf{R}\tilde{\mathbf{X}}_{sfm}. \tag{4.11}$$

We express the initial scale as:

$$s_I = \frac{\sum_{X_i \in \tilde{\mathbf{X}}_{slam}, X_j \in \tilde{\mathbf{X}}_{aligned}} X_i \cdot X_j}{\sum_{X_j \in \tilde{\mathbf{X}}_{sfm}} \|X_j\|^2}, \tag{4.12}$$

where the $\tilde{\mathbf{X}}_{sfm}$ and $\tilde{\mathbf{X}}_{slam}$ are the zero-centered 3D points stacked in a column matrix, and $\otimes$ denotes the outer product between two matrices. Since the depth image is subject to noise, the initial scale s_I is not entirely accurate. We refine the scale in the optimization as shown in Fig. 4.2.

4.2 Optimization and Surfel-based Model Reconstruction

Given the keyframe pose prior and the odometry are initialized and aligned, we optimize the camera poses using the incoming odometry. When a new odometry measurement is obtained using the method in Sec. 4.1.1, it is inserted into the factor graph and incrementally optimized using iSAM2 [40]. Similar to ElasticFusion, the model is generated by surface splatting. Different from the surface deformation approach in ElasticFusion, our method relies on the globally consistent pose priors to constrain the input odometry. Therefore, we do not apply the surface loop closure and the model deformation.

4.3 Experimental Results

4.3.1 Implementation

We implement the system based on two state-of-the-art systems, COLMAP and ElasticFusion. COLMAP is used as the SfM pipeline that generates the pose prior, and the tracking component

in ElasticFusion is used to generate odometry. In addition, we adopt the surfel representation and the surface fusion strategy in ElasticFusion. We implement the proposed factor graph, Levenberg-Marquardt optimizer, and iSAM2 [40] optimizer using the GTSAM [41] library. Experiments are conducted on a Ubuntu 16.04 desktop with an Intel i7-6700 CPU and a GeForce GTX 1070 GPU.

4.3.2 Synthetic Dataset

We test our system on the Augmented ICL-NUIM (A-ICL) dataset with synthetic noise created by Choi et al. [8]. The A-ICL sequence is based on the original ICL-NUIM dataset by Handa et al. [42]. Both ICL and A-ICL sequences are created in the same synthetic living room scene, but A-ICL contains a longer camera trajectory (max of 2870 frames vs. max of 1510 frames) and a larger coverage of the scene. We compare the surface reconstruction accuracy against the ground truth (GT) model using CloudCompare [43], and we also compare the Root Mean Square of Absolute Trajectory Error (ATE-RMSE) using the tools from TUM-RGBD benchmark [44]. To demonstrate the performance of our system, we compare against the systems from three different categories: (1) offline SfM-MVS systems represented by COLMAP-MVS [16], (2) offline RGB-D reconstruction systems represented by Redwood [8], and (3) online dense mapping systems represented by ElasticFusion [4] and InfiniTAMv3 [27]. The surface reconstruction accuracy, system runtime, and absolute trajectory error are used as metrics to evaluate the system performance. Table 4.1 shows the results of the A-ICL living room dataset with noise. A heatmap visualization of the surface reconstruction accuracy is shown in Fig. 4.4. A complete reconstructed model can be view in Fig. 4.6a.

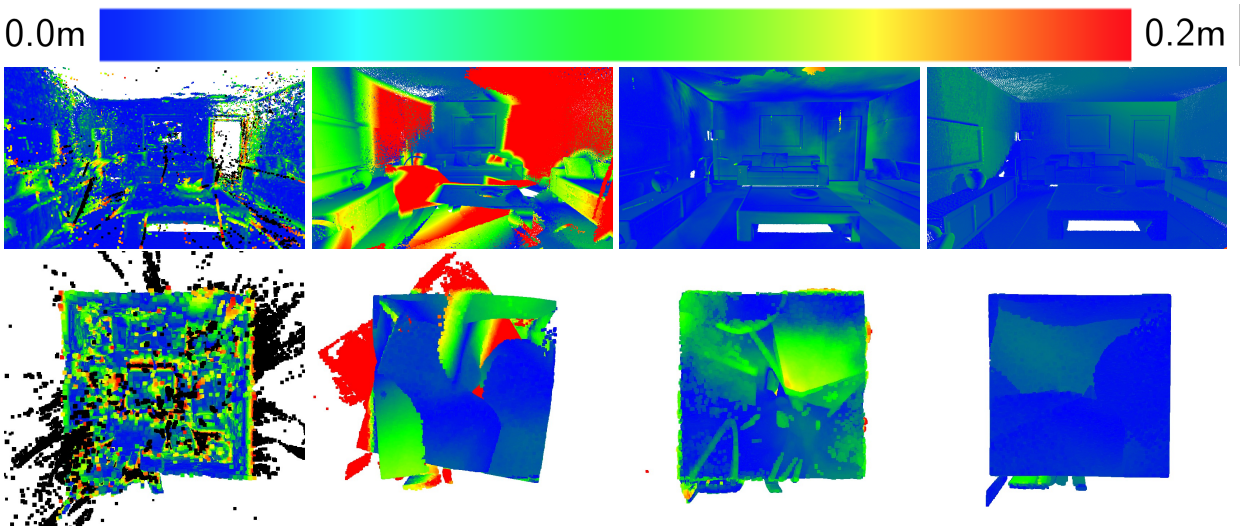
Table 4.1 shows the mean and the standard deviation of the distance to the GT model. Although our surface reconstruction accuracy is slightly lower than the models from COLMAP-MVS and the Redwood, the model produced by our method has the lowest standard deviation and trajectory error. A collection of heatmaps illustrating the surface reconstruction accuracy of A-ICL-lr1 is shown in Fig. 4.4. Furthermore, our method not only achieves a higher accuracy than the online methods, but also finishes the reconstruction in a much shorter period comparing to other offline methods.

Table 4.1: Surface reconstruction accuracy on the synthetic A-ICL dataset

	Mean distance to GT model (cm)		Std. Dev. distance to GT model (cm)	
	A-ICL-lr1	A-ICL-lr2	A-ICL-lr1	A-ICL-lr2
COLMAP-MVS	**1.59**	**1.35**	4.58	4.20
Redwood	3.00	2.02	2.98	1.83
ElasticFusion	7.71	7.78	6.91	6.34
InfiniTAMv3	7.33	10.25	6.39	6.87
Proposed (offline)	1.74	2.26	**1.34**	**1.78**

Table 4.2: System runtime and camera trajectory on the synthetic A-ICL dataset

	System Runtime (min)		ATE RMSE (cm)	
	A-ICL-lr1	A-ICL-lr2	A-ICL-lr1	A-ICL-lr2
COLMAP-MVS	212.66	159.41	5.68	18.78
Redwood	∼300	∼300	N/A	N/A
ElasticFusion	(online)	(online)	66.61	28.53
InfiniTAMv3	(online)	(online)	46.07	73.64
Proposed (offline)	**29.37**	**6.93**	**5.14**	**3.79**



(a) COLMAP (b) ElasticFusion (c) Redwood (d) Proposed

Figure 4.4: Heatmaps showing reconstruction error on A-ICL-lr1. The top row and the bottom row are the internal view and the top view of the room scene, respectively. Points more than 0.2m away from the ground truth are removed.

4.3.3 Real World Dataset

Table 4.3: Results on the real-world CoRBS dataset

	Mean distance (cm)		Std. dev. distance (cm)	
	Racing car	Human	Racing car	Human
ElasticFusion	6.07	52.98	4.45	49.0
Proposed (offline)	**1.30**	**1.12**	**1.15**	**1.84**

We conduct additional tests on the CoRBS [45] real world dataset with surface ground truth model collected by a 3D scanner with sub-millimeter accuracy. We also conduct qualitative evaluation on the TUM RGB-D dataset [44]. The CoRBS dataset is collected using a Kinect V2 hand-held RGB-D camera, and it contains a real racing car and a human-size model, while the TUM RGB-D dataset is collected using a Kinect RGB-D camera with ground truth trajectory. The surface

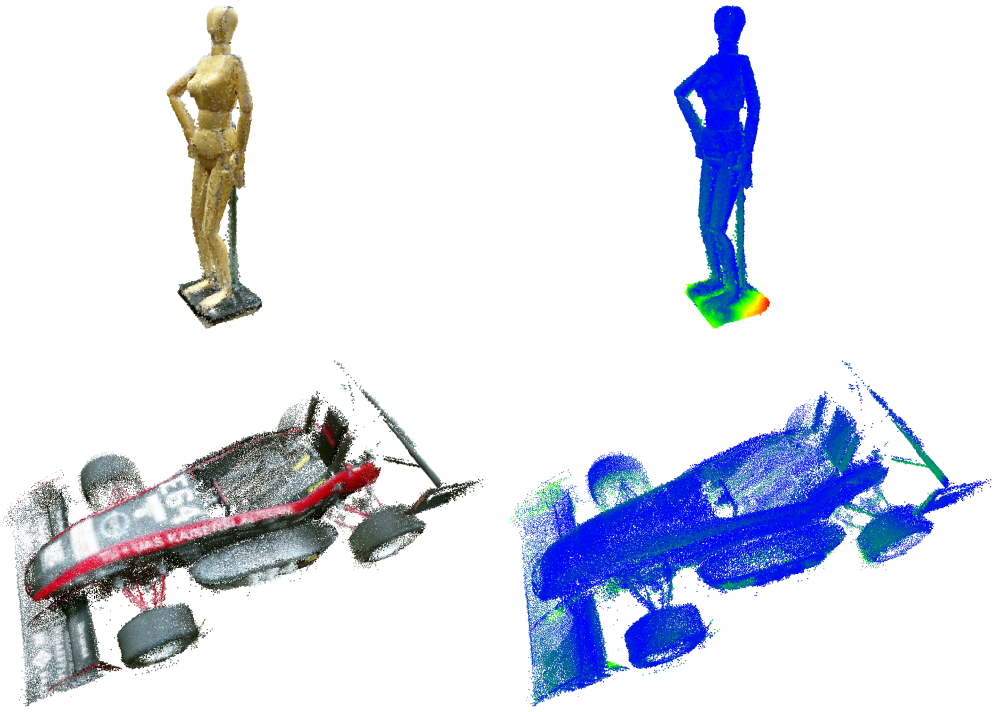


Figure 4.5: Results on the real-world CoRBS dataset. Top row, *Human* sequence. Bottom row, *Racing Car* sequence. The output distance heatmap shows our method reconstructs accurate models.

reconstruction accuracy of each object is shown in Table 4.3, and the reconstructed models and the corresponding heatmaps are shown in Fig. 4.5. In addition, a qualitative view of the TUM RGB-D long office scene [44] can be view in Fig. 4.6b.

4.4 Analysis

We present a dense reconstruction system that combines the advantages from both SfM and SLAM to recover both locally and globally consistent 3D models using a RGB-D sensor. We achieve performance by extracting keyframes and obtaining their corresponding pose priors from SfM, and locally adjusting the camera using odometry from dense SLAM. Our method shows better performance than the state-of-the-art dense SLAM system for both synthetic and real-world datasets, while providing much shorter processing time and comparable quality than other offline dense 3D reconstruction systems. In addition to the discussion about SfM and SLAM, the idea of combining strong camera priors and odometry can also be applied in situations such as GPS input as prior, or IMU input as odometry.

For future work, we would like to develop a solution to various parameter choices. For example, the number of keyframes should be automatically adjusted when the reconstruction data becomes

(a) Augmented-ICL-lr1 [8] top view via splatted rendering

(b) TUM fr3 [44] point cloud office scene

Figure 4.6: Reconstructions results for a synthetic and a real sequence

larger, and more keyframes should be extracted when the camera motion is large. In addition, it is also of great interest to build an online system that is capable of achieving similar reconstruction accuracy. In Chap. 5, we will introduce our design and efforts in building such a system.

Chapter 5

Elastic Structure Fusion: Dense Mapping with Deformation and Pose Graph

Chapter 4 demonstrates the capability of our offline system in reconstructing a large scale 3D map. However, our offline system is still highly dependent on the SfM system to provide good *keyframe* pose priors. It is not always possible for the SfM system to successfully extract the keyframe prior (for example, SfM often struggles to perform well in featureless regions), and without the priors our proposed system degenerates into the dense SLAM formulation. In addition, if odometry drifts significantly between two keyframes, the model could become distorted due to rendering from the frames in between. Finally, a natural question that arises from the previous work is whether it is possible to achieve real-time performance with the current reconstruction system. To solve this problem, we propose a new online system that leverages the deformation to link together the surface model correction and the camera pose estimation.

5.1 Approach Overview

Our system derives from a typical dense SLAM formulation that leverage the fast and parallel computation on GPU. The system has three different components: tracking, mapping, and loop closure. For tracking and surfel fusion mapping, we adapt a tracking framework similar to Elastic-Fusion [4]. In addition, we add the feature points and landmark creation on CPU to better estimate the camera poses. The result from the sparse feature and landmark tracking is used to initialize the dense tracking. As a result, the map is composed of two different layers: a sparse map represented by the feature-based landmark, and a dense map from the surfel fusion. Upon loop closure, the sparse layer is used to construct a deformation graph that is embedded in the surface of the model, and the deformation graph is used to guide the deformation of the dense model in order to recover global and local geometry. With the two-layer architecture, we successfully link the pose graph and deformation graph and allow model deformation based on the change in the camera poses. Fig.

4.1 shows the overall system structure, and Fig. 5.4 shows the structure of our new loop closure component.

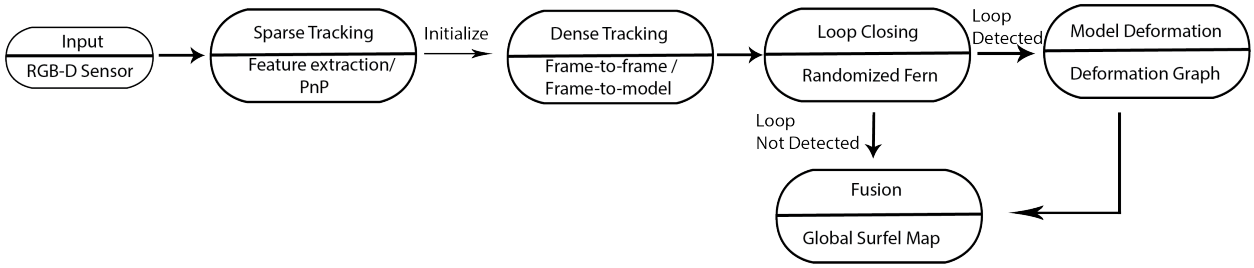


Figure 5.1: System diagram. Given a stream of RGB-D images, we first extract features and conduct sparse feature tracking using conventional algorithms such as PnP. Then, we use the result from PnP to initialize the dense frame-to-model or frame-to-frame tracking. If a loop closure is detected by the randomized fern, a deformation graph will be constructed based on the results of bundle adjustment upon loop closure. Then, the constructed deformation graph will be uploaded to GPU and deform the surfels.

5.2 Joint Geometric and Photometric Tracking

We refer the reader to 3.2 for details in the camera tracking.

5.3 Loop Closure

We implement the loop closure detection using the randomized fern proposed by Glocker et al. [46]. Once two similar images are detected, we try to align two images using joint geometric and photometric odometry. Then, we check both the number of ICP inliers and residuals. If the number of inliers is larger than 50% of the current frame's point number and residual is smaller than the threshold ϵ_{icp}, we claim that the loop closure is valid. The overview of the loop closure structure is shown in Fig. 5.2.

5.3.1 Randomized Fern Relocalization

We use the randomized fern to detect loop closures. Our implementation of the randomized fern is similar to that of the Glocker et al. [46]. We define a fern $F = \{f_i\}_{i=1}^{n}$ to be a set of consecutive nodes, where each node represents a binary test of a particular pixel location θ_i and a value τ_i. The binary test can be described as:

$$f(I, \theta, \tau) = \begin{cases} 1 \text{ if } I(\theta) \geq \tau \\ 0 \text{ if } I(\theta) < \tau, \end{cases} \tag{5.1}$$

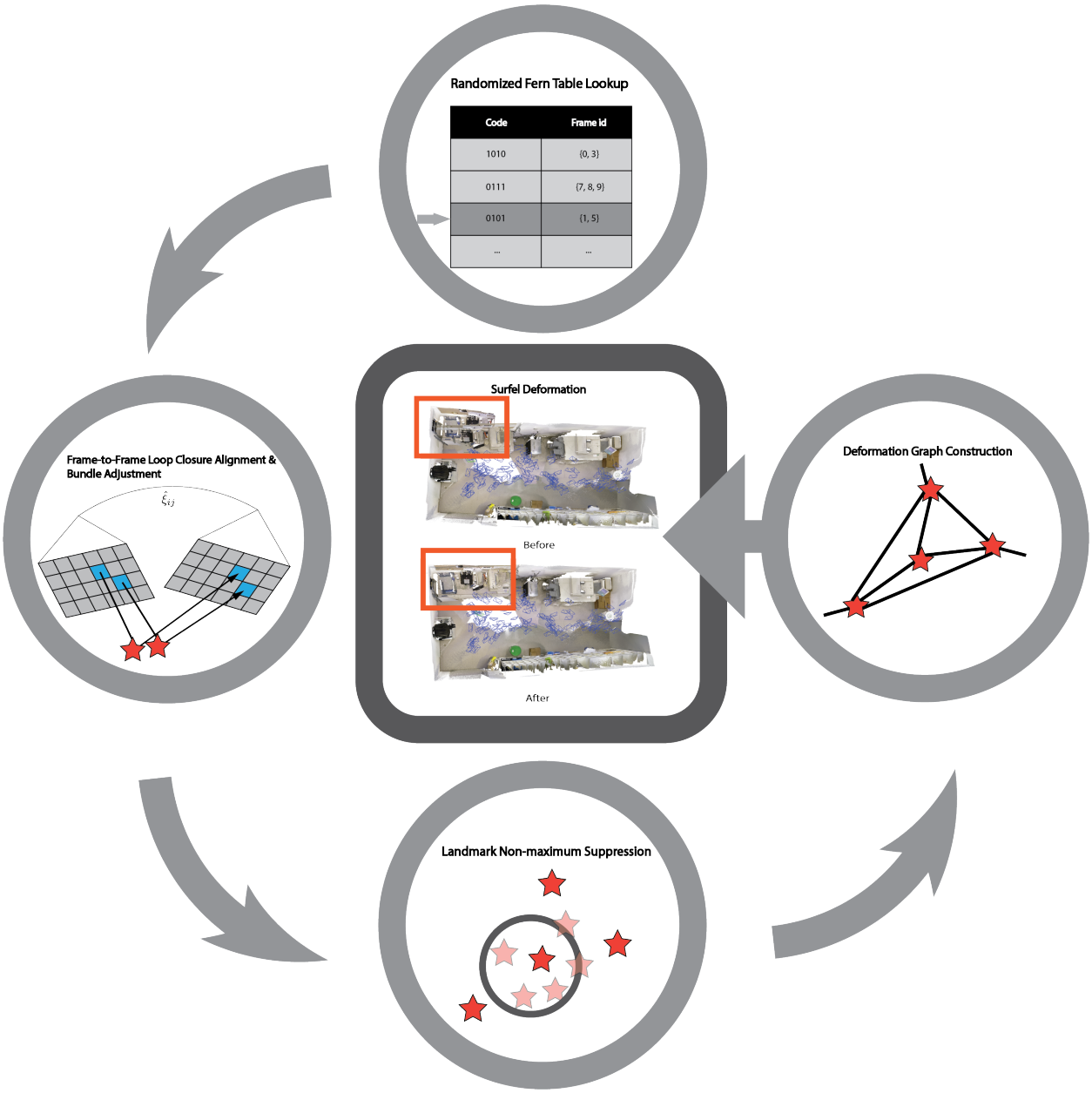


Figure 5.2: Loop closure system diagram

where τ is a threshold on the pixel value $I(\theta)$. By applying the nodes to the pixel value we obtained a binary code block with n binary values $b_1 b_2 ... b_n \in \mathbb{B}^n$. We called the collection of ferns a conservatory such that $C = \{F_i\}_{i=1}^{m}$. Each conservatory yields a single code block $b_c = b_{F_1} b_{F_2} ... b_{F_m} \in \mathbb{B}^{mn}$. Each code block represents the binary encoding of a single image. We define the Hamming distance between two different code blocks from frame I and frame J to be as

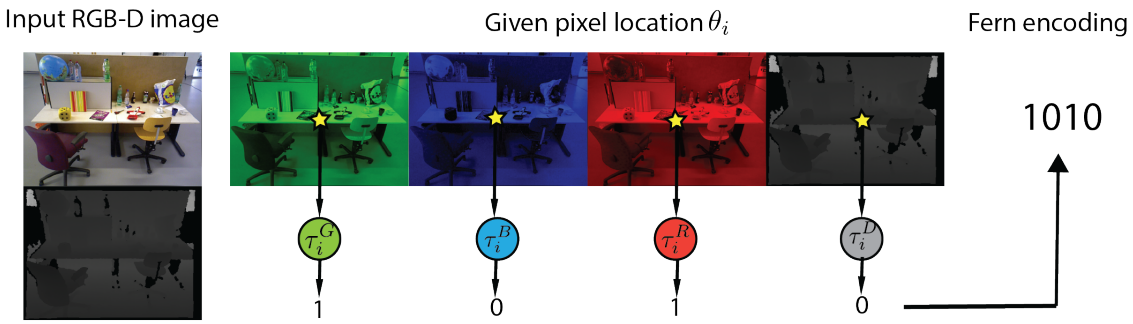


Figure 5.3: Fern encoding example. Given a pair of RGB-D image, we randomly choose a pixel location and RGB-D thresholds, and then we evaluate the pixel value on the thresholds using Eqn. 5.1. This results to a 4 bit encoding of the frame.

$$\text{BlockHD}(b_C^I, b_c^J) = \frac{1}{m}\sum_{k=1}^{m} b_C^I \equiv b_C^J \tag{5.2}$$

The $\equiv$ sign gives 0 if two code blocks are completely equal, and 1 if there is at least one bit difference. This gives a dissimilarity score between two frames in the interval of $[0, 1]$.

To construct ferns for each keyframe image, we first downsample the input image to 60×80. Then, we randomly choose 500 fern locations that are uniformly distributed across the downsampled image space. Each fern consists of 4 nodes where $f = \{f_R, f_G, f_B, f_D\}$. This also gives us 4 parameters $\{\theta_R, \theta_G, \theta_B, \theta_D\}$. Following the guideline from Glocker et al. [46], we pick our 4 parameter by randomly choosing from the input domain, which gives us $\{\tau_R, \tau_G, \tau_B\} \in [0, 255]$, and $\{\tau_D\} \in [30, 3000]$ given that the valid depth input range is between 30 mm and 3000 mm.

For every k frames, we extract a keyframe and its corresponding tracked pose $\{I, T\}_{id}$ where $T \in \text{SE}(3)$ from the image sequence and calculate its code representation b_C^I. We store its id and its keyframe and pose into a table, where id can be used to lookup the keyframe image and poses. In addition, we store the keyframe id and its code block into a set P. For every incoming new frame, we simply calculate the co-occurrence of all of the previously stored keyframe. Once we found the minimum BlockHD dissimilarity score and this score is below a given threshold σ, we extract such minimum keyframe image and pose from the table where it is stored. Then, we initiate an odometry instance to try to align the keyframe and the newly incoming frame. The loop closure is deemed valid if the number of ICP inliers n_{icp} is above the threshold n_{icp}^t and per-vertex residual ϵ_{icp} is below the threshold ϵ_{icp}^t. Both thresholds are user-defined hyperparameters.

5.3.2 Pose Graph Optimization

We maintain a pose factor of of each frame. Upon a valid loop closure is detected, a loop closure factor is added into the factor graph, and iSAM2 algorithm [40] is used to optimize the pose graph.

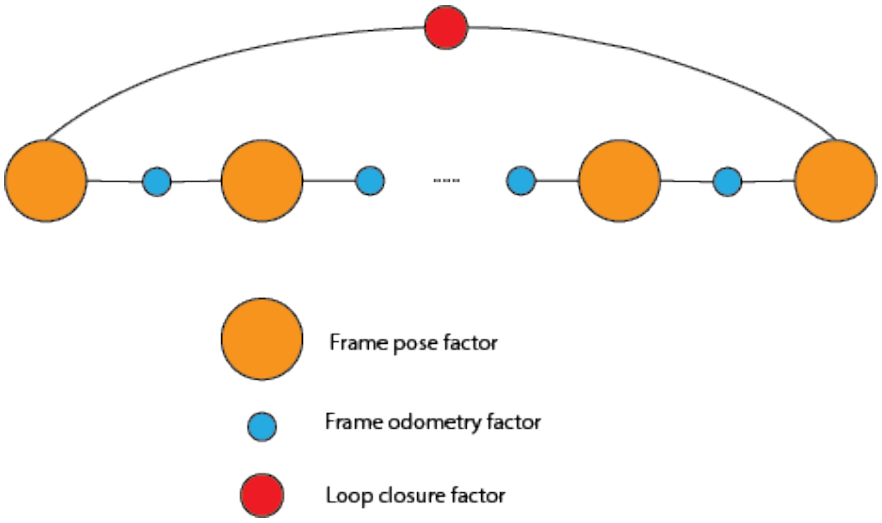


Figure 5.4: Pose graph optimization upon loop closure

The loop closure factor is composed by the odometry alignment of the keyframe from P and the newly incoming frame. The factor graph construction is shown in Fig. 5.4.

5.3.3 Retriangulation and Bundle Adjustment

When pose graph is updated due to loop closure, we first re-triangulate all landmarks in the map. Given a landmark $\mathbf{X}_j$ and a set of corresponding frames I that observes this landmark, we have the new landmark position $\tilde{\mathbf{X}}_j$ to be:

$$\tilde{\mathbf{X}}_j = \frac{1}{n_f} \sum_{i}^{n_f} \pi^{-1}(\mathbf{K}, \mathbf{P}_i, \mathbf{x}_{ij}), \tag{5.3}$$

where $\mathbf{K}$ is the intrinsics matrix, $\mathbf{P}_i$ is the pose of the ith frame, and $\mathbf{x}_{ij}$ is the measurement of the jth landmark in the ith frame. In addition, we check whether if the projection error is too large among different observation. If two frames show that they have very distinctive measurements on the same landmark, it is very likely that this landmark is mistakenly associated with another landmark. As a result, we discard this landmark from the list of all landmarks in an effort to remove the outliers.

After we geometrically verify the landmark correspondence, we conduct a step of bundle adjustment that jointly optimize all landmarks and frame poses. We have:

$$\arg\min_{\mathcal{X},\mathcal{P}} \sum_{\mathbf{P}_i \in \mathcal{P}} \sum_{\mathbf{X}_j \in \mathcal{X}} \rho(E_{BA}) \tag{5.4}$$

where E_{BA} is defined in Eq. 3.1. This step make sure that we reach the optimal state for all frames and landmarks after the loop closure.

5.3.4 Feature Non-maximum Suppression

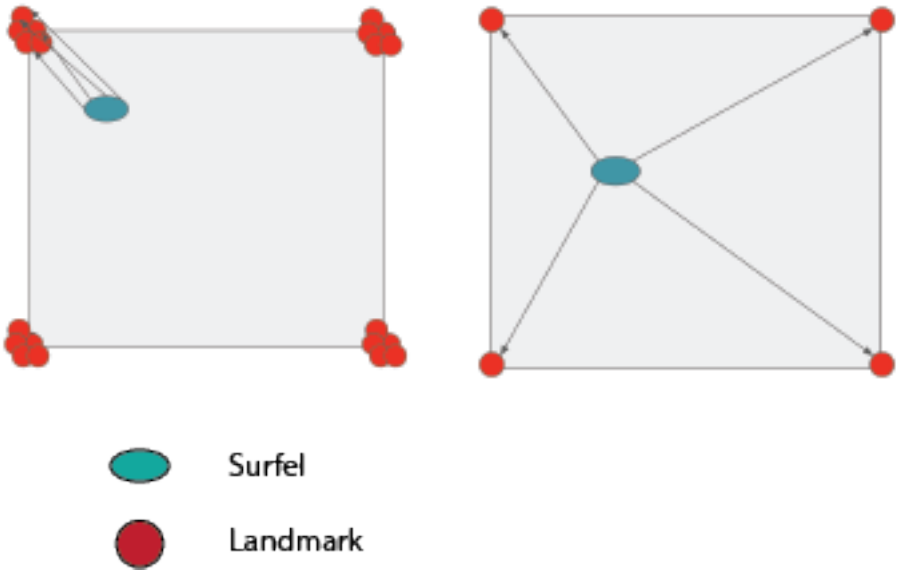


Figure 5.5: Landmark clustering and non-maximum suppression. In order to effectively apply deformation and exclude the effect from multiple concentrated landmarks, we conduct a non-maximum suppression based on the number of observations. Previously a surfel is deformed by landmarks at one single corner, and after non-maximum suppression, the surfel is deformed by landmarks that are dispersed around the region.

Feature extraction is one of the most commonly used techniques in image processing. Features on an image usually concentrates on the *corner* regions, where two distinctive lines meet. These corner regions usually represents a high image gradient. Since image features often concentrates on these regions, it is not uncommon to observe multiple image features clustered in a single region. Since the landmarks are constructed from the image features, we can often observe multiple landmarks clustered in the corner regions, as shown in the left part of Fig. 5.5. Given that we are using the landmarks as the skeleton of the deformation, we want the surfels to be stretched equally across all directions, instead of all from landmarks situated in a single corner. Therefore, we apply non-maximum suppression to reduce a cluster of landmarks into a single landmark.

In order to cluster the landmarks, we used the k-d tree data structure to efficiently indexing the spatial locations of the landmarks. The complete non-maximum suppression algorithm is listed in

Alg. 1.

Algorithm 1: Landmark non-maximum suppression

Data: $\mathcal{X}$, a set of all landmarks
Result: $\mathcal{C}$, a set of clustered (set) of landmarks

Visited $\longleftarrow$ A list of the same size of landmarks initialize with boolean false;
$\mathcal{C} \longleftarrow \emptyset$
foreach *Landmark* $\mathbf{X}_j$ *in* $\mathcal{X}$ **do**
  if *Visited(*$\mathbf{X}_j$*)* **then**
    continue;
  end
  $\mathbf{C} \longleftarrow \emptyset$;
  $Q \longleftarrow$ Empty Queue;
  $Q.push(\mathbf{X}_j)$;
  $\mathbf{C}.push(\mathbf{X}_j)$;
  Visited($\mathbf{X}_j$) = true;
  while *not* Q*.empty()* **do**
    $\mathbf{X}_j = Q.pop()$;
    find k nearest neighbor of the landmark $\mathbf{X}_j$ using KD-tree;
    foreach *neighbor landmark* $\mathbf{X}_{jk}$ **do**
      Mark corresponding Visited as true;
      $\mathbf{C}.push(\mathbf{X}_{jk})$;
    end
  end
  $\mathcal{C}.push(\mathbf{C})$;
end

For each cluster $\mathbf{C}_i$ in the set of all clusters $\mathcal{C}$, we find the landmark with the maximum number of observations. If more than one landmark have the same number of observations, a random landmark is chosen. This landmark will be used to represent the entire cluster of landmarks, such that the reset of the landmarks in the cluster are discarded. The resulting landmark is shown in the right figure in Fig. 5.5.

5.3.5 Deformation Graph using Landmarks

The deformation graph is used to non-rigidly deform the surface to preserve model's global and local consistency. Since it is computationally prohibitive to conduct bundle adjustment across millions of surfels, we develop a way to indirectly manipulate the surface and add constraints upon loop closure. We use the features and landmarks as an intermediate step to conduct shape manipulation while preserving the shape integrity. Our deformation graph formulation is similar to that of Sumner et al. [47] and Whelan et al. [4].

A deformation graph is a graph structure that consists of a set of nodes and edges. The nodes are represented by the landmarks, and each node $\mathcal{G}^n$ consist of a timestamp $\mathcal{G}^n_t \in \mathbb{R}$, a position vector $\mathcal{G}^n_{\mathbf{g}} \in \mathbb{R}^3$, a 3×3 matrix $\mathcal{G}^n_{\mathbf{R}}$, and a 3×1 translation vector $\mathcal{G}^n_{\mathbf{t}}$. The matrix $\mathcal{G}^n_{\mathbf{R}}$ and the translation vector $\mathcal{G}^n_{\mathbf{t}}$ make up an affine transformation, which is default to be an identity transformation. Each node is also connected to k_g neighbors. Upon loop closure, the $\mathcal{G}^n_{\mathbf{R}}$ and $\mathcal{G}^n_{\mathbf{t}}$ are optimized according to the following energy formulation:

$$E_{def} = w_{rot}E_{rot} + w_{reg}E_{reg} + w_{con}E_{con}, \tag{5.5}$$

given a set of landmark correspondence before and after triangulation and bundle adjustment. We define the landmark correspondence before as $\mathbf{X}^s_j$ and after as $\mathbf{X}^e_j$. We detail each term of the energy function below:

- *Rotation constraint*: For the deformation, we want to the rotation term in the affine transformation to be as rigid as possible. Therefore, we have

$$E_{rot} = \sum_j \left\| \mathcal{G}^n_{\mathbf{R}} (\mathcal{G}^n_{\mathbf{R}})^\top - I \right\|^2_F, \tag{5.6}$$

  where it uses the Frobenius norm to sum up the term.

- *Regularization constraint*: For the deformation, we want to make sure that the surface to be smooth and without obvious cracks and gaps. We use the following formulation to optimize the smoothness of the surface:

$$E_{reg} = \sum_j \sum_k \left\| \mathcal{G}^n_{\mathbf{R}} (\mathcal{G}^k_{\mathbf{g}} - \mathcal{G}^n_{\mathbf{g}}) + \mathcal{G}^n_{\mathbf{g}} + \mathcal{G}^n_{\mathbf{t}} - \mathcal{G}^k_{\mathbf{g}} - \mathcal{G}^k_{\mathbf{t}} \right\|^2_2. \tag{5.7}$$

- *Landmark constraint*: We must ensure the position of the landmarks after the triangulation and bundle adjustment should be as close to the target location as possible. Therefore, we have

$$E_{con} = \sum_j \left\| \phi(\mathbf{X}^s_j) - \mathbf{X}^e_j \right\|^2_2, \tag{5.8}$$

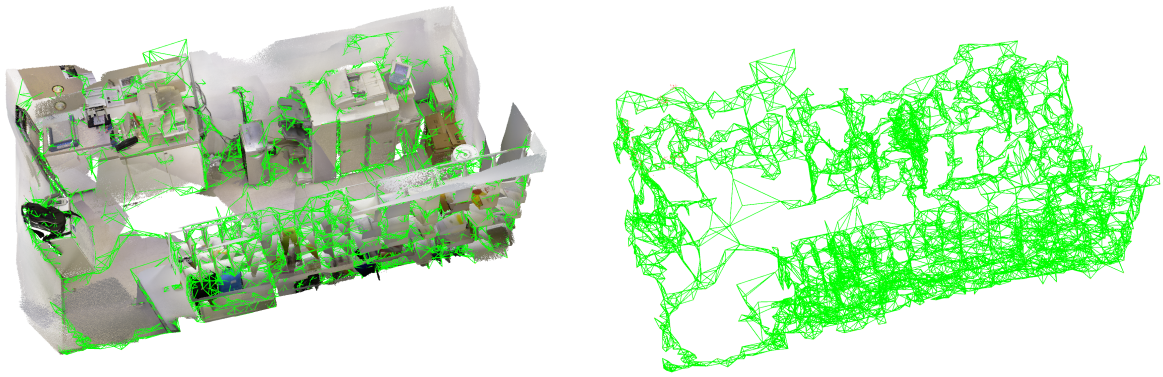
  where $\phi(\cdot)$ can be described as the influence from the neighboring deformation nodes

$$\phi(\mathbf{X}^s_j) = \sum_n^{k_g} w^n(\mathbf{X}^s_j)(\mathcal{G}^n_{\mathbf{R}}(\mathbf{X}^s_j - \mathcal{G}^n_{\mathbf{g}}) + \mathcal{G}^n_{\mathbf{g}} - \mathcal{G}^n_{\mathbf{t}}), \tag{5.9}$$

  and the weight $w^n(\cdot)$ is calculated as:

$$w^n(\mathbf{X}^s_j) = (1 - \left\| \mathbf{X}^s_j - \mathcal{G}^n_{\mathbf{g}} \right\| / d_{max})^2, \tag{5.10}$$

  where d_{max} is the distance to the $k_g + 1$ node. We use $k = 4$ in all of our experiments aligned with the suggestion from Sumner et al. [47]. In addition, we use $w_{rot} = 1$, $w_{reg} = 10$, and $w_{con} = 100$ in all our experiments.

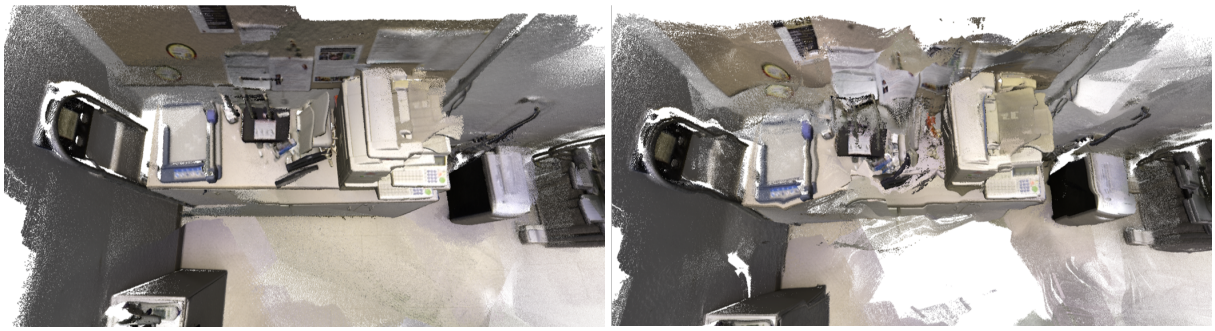


(a) Surface reconstruction model overlay with deformation graph

(b) Embedded surface deformation graph

Figure 5.6: An example of the deformation graph in the *copyroom* [20] dataset

We introduce the steps to construct a deformation graph. Similar to ElasticFusion, the process of applying the deformation graph to the surfels is introduced in Alg. 2.



(a) Before applying deformation

(b) After applying deformation

Figure 5.7: Deformation based on landmarks. The picture on the left shows the before the deformation, and the picture on the right shows the after deformation. Although the multiple mapping layers has been corrected, the local consistency is no longer preserved due to the landmarks being placed to the wrong locations in the process of bundle adjustment.

However, after implementing the landmark-based deformation graph, we found that the resulting map suffers from local consistency issues. An example of this issue can be seemed in Fig. 5.7. We found that the most likely underlying cause for such problem is the incorrect data association of landmarks, which leads to the landmarks being placed in the wrong location in the bundle adjustment process. While using some heuristics might mitigate the problem of incorrect data association of the image features, due to the inherent limitation of feature matching process and the difficulty of the data association, it is a challenging problem to ensure the correct data association of the landmarks. Alternatively, we proposed to use the camera poses as the deformation node instead to avoid directly addressing this problem.

Algorithm 2: Deformation graph application

input : $\mathcal{M}^s$ surfels to be deformed
$\mathcal{G}$ a set of deformation nodes
α number of nodes to explore
output: $\hat{\mathcal{M}}^s$ deformed surfels

// Gather temporally nearby nodes and sort them by euclidean distances

$c \leftarrow \arg\min_i \left\| \mathcal{M}^s_{t_0} - \mathcal{G}^i_{t_0} \right\|_1$

$\mathcal{I} \leftarrow \emptyset$

for $i \leftarrow -\alpha/2$ **to** $\alpha/2$ **do**

  $\mathcal{I}^{i+\alpha/2} \leftarrow c + i$

sort by euclidean distance($\mathcal{I}$, $\mathcal{G}$, $\mathcal{M}^s_p$)

take closest k as influencing nodes

// Calculate weights

$h \leftarrow \mathbf{0}$

$d_{\max} \leftarrow \left\| \mathcal{M}^s_{\mathrm{p}} - \mathcal{G}^{\mathcal{I}^k}_{\mathrm{g}} \right\|_2$

for $n \in \mathcal{I}\left(\mathcal{M}^s, \mathcal{G}\right)$ **do**

  $w^n\left(\mathcal{M}^s\right) \leftarrow \left(1 - \left\| \mathcal{M}^s_{\mathrm{p}} - \mathcal{G}^n_{\mathrm{g}} \right\|_2 / d_{\max}\right)^2$

  $h \leftarrow h + w^n\left(\mathcal{M}^s\right)$

// Apply transformation

$\hat{\mathcal{M}}^s_{\mathbf{p}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} \left[\mathcal{G}^n_{\mathbf{R}} \left(\mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^n_{\mathbf{g}}\right) + \mathcal{G}^n_{\mathbf{g}} + \mathcal{G}^n_{\mathbf{t}}\right]$

$\hat{\mathcal{M}}^s_{\mathbf{n}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} \mathcal{G}^{n-1}_{\mathbf{R}} \top \mathcal{M}^s_{\mathbf{n}}$

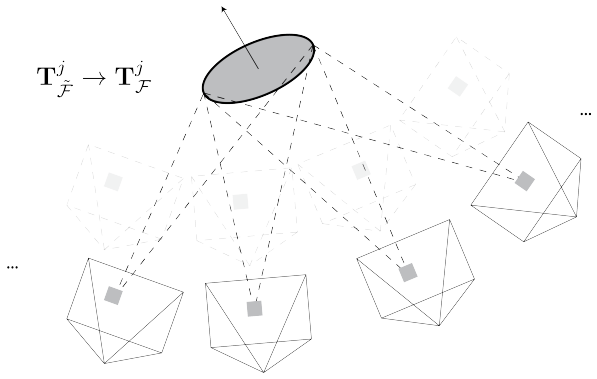


Figure 5.8: Each surfel is associated with k frames that most recently observe such surfel.

5.3.6 Deformation Graph using Camera Poses

An important observation from the factor graph structure of the SLAM problem shows that the number of camera poses are usually much fewer than the number of the landmarks, given that hundreds of features will be extracted from a single image. As a result, camera poses are much more well-constrained comparing to the landmarks, and using camera poses instead of landmarks as deformation node becomes a natural alternative choice. Using the same factor graph framework, we replace the deformation node with landmarks, and using the pose before bundle adjustment $\mathbf{T}^j_{\tilde{\mathcal{F}}}$ and the pose after bundle adjustment $\mathbf{T}^j_{\mathcal{F}}$ as the deformation parameter. In addition, we associate each surfel with k frames that most recently observe such surfel, as shown in Fig. 5.8. Our new deformation application can be written as follows:

$$\hat{\mathcal{M}}^s_{\mathbf{p}} = \exp\left(\frac{1}{k}\sum_j^k \log\left((\mathbf{T}^j_{\tilde{\mathcal{F}}})^{-1}\mathbf{T}^j_{\mathcal{F}}\right)\right)\mathcal{M}^s_{\mathbf{p}}, \tag{5.11}$$

$$\hat{\mathcal{M}}^s_{\mathbf{n}} = \exp\left(\frac{1}{k}\sum_j^k \log\left((\mathbf{T}^j_{\tilde{\mathcal{F}}})^{-1}\mathbf{T}^j_{\mathcal{F}}\right)\right)\mathcal{M}^s_{\mathbf{n}}, \tag{5.12}$$

where the surfel position and normal are transformed by the average transformation of $(\mathbf{T}^j_{\tilde{\mathcal{F}}})^{-1}\mathbf{T}^j_{\mathcal{F}}$ across the k frames that observe the surfel. For the simplicity, we assume both the position and the normal vector are of the homogeneous form.

Fig. 5.9 shows the resulting deformation using camera poses instead of the landmarks. The result shows that we are able to preserve both the local and global consistency in our new deformation formulation.

Figure 5.9: The deformation using the camera pose as the new deformation nodes. Note that comparing to Fig. 5.7a, we preserve both the local and global consistency.

5.4 Experimental Results

5.4.1 System Information

We implement and test our system in Ubuntu 16.04 desktop with an Intel i7-6700 CPU. We test using both CUDA 9.0 and CUDA 10.0 on a GeForce GTX 1070 GPU, as well as a GeForce GTX 2070 GPU. The surfel fusion, rendering, and odometry are implemented using CUDA in GPU. The pose and landmark bundle adjustment, as well as the deformation graph optimization, are implemented in CPU. Specifically, the pose and landmark optimization are represented using a factor graph and optimized using GTSAM [41].

5.4.2 Synthetic Dataset

Similar to the offline system, we conduct our surface reconstruction accuracy test on the A-ICL [8] dataset with synthetic noise. Table 5.1 shows the surface reconstruction result on the A-ICL living room scene. The result shows that our system achieve superior performance on par with the other online systems, while still needs to improve comparing to the offline systems.

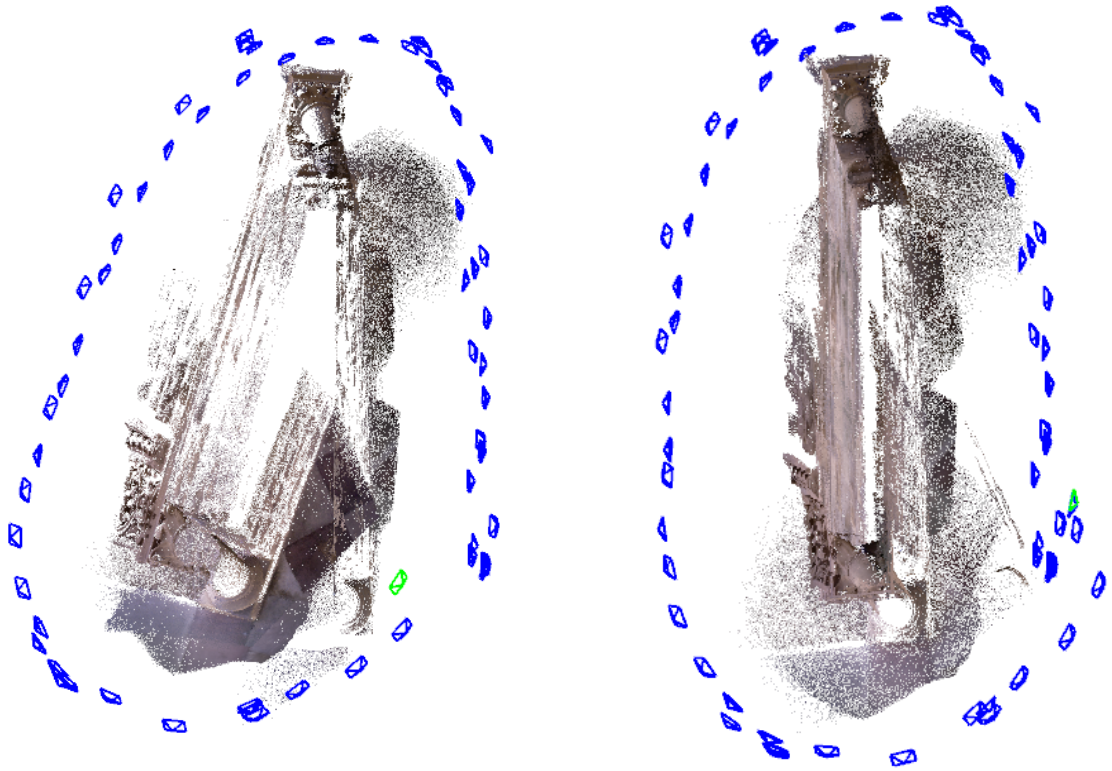
5.4.3 Real World Dataset

We also conduct qualitative evaluation on many real world dataset. Mainly, we conduct our experiments on the publicly available *stanford* dataset [20] and *redwood* dataset [48]. Fig. 5.10 shows the effect of deformation that successfully corrects the odometry drift after loop closure. Fig. 5.11 shows the reconstruction of a totempole outdoor that contains a major loop closure. In addition, Fig. 5.12 shows the reconstruction of a hotel lobby that contains 20,000 frames and covering an area of 86.46m^2. Fig. 5.13 shows the reconstruction of the *The Burghers of Calais* statues. This

Table 5.1: Surface reconstruction accuracy on the synthetic A-ICL dataset

	Mean distance to GT model (cm)		Std. Dev. distance to GT model (cm)	
	A-ICL-lr1	A-ICL-lr2	A-ICL-lr1	A-ICL-lr2
COLMAP-MVS	**1.59**	**1.35**	4.58	4.20
Redwood	3.00	2.02	2.98	1.83
Proposed (offline)	1.74	2.26	**1.34**	**1.78**
ElasticFusion	7.71	7.78	6.91	6.34
InfiniTAMv3	7.33	10.25	6.39	6.87
Proposed (online)	2.28	5.64	3.60	4.15

dataset is particularly challenging given that it is collected outdoor with interference from the sunlight, and it also suffers from significant motion blur. These real-world dataset shows that our system can perform well in mid-size to large scale mapping tasks, and generate reliable 3D model under imperfect sensor conditions.



(a) *stonewall* dataset before deformation

(b) *stonewall* dataset after deformation

Figure 5.10: Reconstruction of the *stonewall* [20] dataset, before and after deformation. Severe odometry drift and the distorted model are corrected after deformation.

Figure 5.11: Reconstruction of the *totempole* [20] dataset

Figure 5.12: Reconstruction of the *lobby* [48] dataset that contains 20000 frames

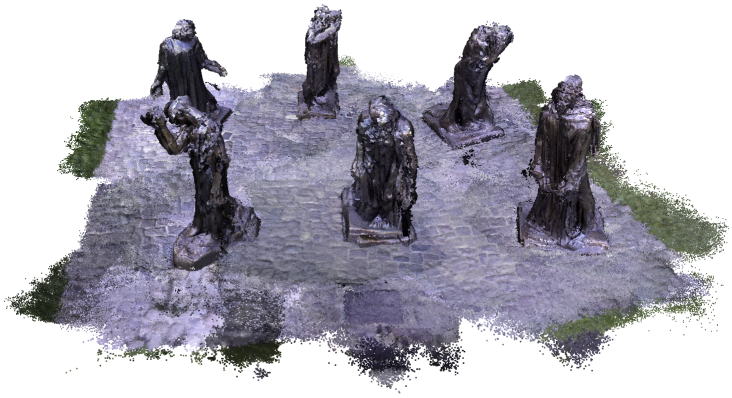


Figure 5.13: Reconstruction of the *burghers* [20] dataset that contains over 10000 frames

5.5 Analysis

We present a dense 3D SLAM system that is capable running online while preserving global consistency through shape deformation. We achieve better reconstruction accuracy comparing to the state-of-the-art systems, while using only one single GPU to conduct simultaneous tracking, mapping, and loop closure. However, our system still suffers from the inaccurate tracking in the front-end, which leads to significant drift in the odometry that has to be corrected in frequent small loop closure. In addition, similar to all appearance-based methods, our fern-based localization does not necessarily address all the possibility of loop closures, especially during noisy depth sensors and drastic lighting changes. These are still the direction that should be improved in the future work.

Chapter 6

Conclusions

We closely examine the problem of dense 3D reconstruction and mapping using a depth-enabled camera. Inspired by the previous work on both sparse and dense SLAM systems, we combine the SfM-MVS pipeline from the computer vision community and the SLAM method from the robotics community to achieve high-fidelity mapping. Specifically, we introduce two 3D reconstruction systems that achieve state-of-the-art performance in constructing a 3D map in surfel representation. The offline system is able to achieve high accuracy mapping comparing to online SLAM systems, while it significantly reduces the computation time. The online system leverages deformation via loop closure to achieve global accuracy, using only a single GPU to conduct mapping, tracking, and model correction. The online system is also open-source for the benefit of the community. Through experiments, we validate the system performance in various scenarios in mapping different environments.

Currently, both the online and offline systems are subject to various of limitations. Specifically, the offline system is still highly dependent on the SfM pipeline. If the SfM pipeline provides wrong estimation of the camera pose prior, the system will suffer from severe drift due to the strong prior introduced into the factor graph. The online system also suffers from significant drift in the front-end due to inaccurate tracking.

One possible solution to improve the tracking is to use the feature point to coarsely estimate the motion of the camera, and then use the joint geometric and photometric tracking to improve upon the result. There are two possible ways: first, we can jointly optimize the result from feature and from dense tracking; second, we can use the result from the feature point to initialize the dense tracking. Both ways could provide better odometry estimation then relying on dense tracking alone.

We believe that there are still many open problems in the area of 3D mapping. For example, how to leverage the scene semantic information and use it the improve robot perception of the world is a problem that can be built upon the constructed dense map. In addition, achieving both local and global accuracy in online dense SLAM is also a very challenging problem. Finally, we would like to address the problem of scalability of mapping given the system memory limitation. All of those

problems are challenging yet rewarding to explore in the future of dense 3D mapping.

Bibliography

[1] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Oct. 2011, pp. 127–136. 1.1, 1.2, 2.4

[2] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Trans. on Graphics*, vol. 32, no. 6, pp. 169:1–169:11, Nov. 2013. [Online]. Available: http://doi.acm.org/10.1145/2508363.2508374 1.1

[3] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended KinectFusion," *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Jul. 2012. 1.1, 1.2, 2.4

[4] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems (RSS)*, Rome, Italy, Jul. 2015. 1.1, 1.2, 2.2, 2.4, 4.3.2, 5.1, 5.3.5

[5] Q. Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3d data processing," *Computing Research Repository*, vol. abs/1801.09847, 2018. [Online]. Available: http://arxiv.org/abs/1801.09847 1.1, 2.2

[6] P. Moulon, P. Monasse, and R. Marlet, "Adaptive structure from motion with a contrario model estimation," in *Asian Conf. on Computer Vision (ACCV)*, Nov. 2012, pp. 257–270. 1.2, 2.1, 4.1.1

[7] J. L. Schönberger, E. Zheng, J. M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Eur. Conf. on Computer Vision (ECCV)*, Oct. 2016, pp. 501–518. 1.2, 2.1, 4.1.1

[8] S. Choi, Q. Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Jun. 2015, pp. 5556–5565. 1.2, 2.2, 4.3.2, 4.6a, 5.4.2

[9] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration," *ACM Trans. on Graphics*, vol. 36, no. 4, May 2017. [Online]. Available: http://doi.acm.org/10.1145/3072959.3054739 1.2, 2.4

[10] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in

3d," *ACM Trans. on Graphics*, vol. 25, no. 3, pp. 835–846, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141964 2.1

[11] ——, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1007/s11263-007-0107-3 2.1

[12] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building rome in a day," *Commun. ACM*, vol. 54, no. 10, pp. 105–112, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/2001269.2001293 2.1

[13] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 32, no. 8, pp. 1362–1376, Aug. 2010. 2.1

[14] P. Moulon and P. Monasse, "Unordered feature tracking made fast and easy," *ACM SIGGRAPH European Conference on Visual Media Production*, p. 1, Dec. 2012. 2.1

[15] C. Sweeney, T. Höllerer, and M. Turk, "Theia: A fast and scalable structure-from-motion library," in *Proceedings of the 23rd ACM Intl. Conf. on Multimedia*, Oct. 2015, pp. 693–696. 2.1

[16] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Jun. 2016, pp. 4104–4113. 2.1, 4.1.1, 4.3.2

[17] L. Moisan, P. Moulon, and P. Monasse, "Automatic homographic registration of a pair of images, with a contrario elimination of outliers," *Image Processing On Line*, vol. 2, pp. 56–73, 2012. 2.1

[18] S. Zhu, T. Fang, J. Xiao, and L. Quan, "Local readjustment for high-resolution 3D reconstruction," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Jun. 2014, pp. 3938–3945. 2.1, 2.1

[19] Y. Yao, S. Li, S. Zhu, H. Deng, T. Fang, and L. Quan, "Relative camera refinement for accurate dense reconstruction," in *Intl. Conf. on 3D Vision*, Oct. 2018, pp. 185–194. 2.1

[20] Q. Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics*, pp. 112:1–112:8, Jul. 2013. 2.2, 5.6, 5.4.3, 5.10, 5.11, 5.13

[21] Q. Y. Zhou, S. Miller, and V. Koltun, "Elastic fragments for dense scene reconstruction," in *Intl. Conf. on Computer Vision (ICCV)*, Dec. 2013, pp. 473–480. 2.2

[22] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Intl. Conf. on Computer Vision (ICCV)*, vol. 2, Oct. 2003, pp. 1403–1410. 2.3

[23] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, Nov. 2007, pp. 225–234. 2.3

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardös, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015. 2.3, 2.4

[25] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM,"

in *Eur. Conf. on Computer Vision (ECCV)*, Sep. 2014, pp. 834–849. 2.3

[26] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray, "Very high frame rate volumetric integration of depth images on mobile device," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 11, pp. 1241–1250, Nov. 2015. 2.4

[27] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray, "InfiniTAM v3: A framework for large-scale 3D reconstruction with loop closure," *ArXiv e-prints*, Aug. 2017. 2.4, 4.3.2

[28] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *SIGGRAPH*, 1996, pp. 303–312. [Online]. Available: http://doi.acm.org/10.1145/237170.237269 2.4

[29] H. L and L. F, "FlashFusion: Real-time globally consistent dense 3D reconstruction using CPU computing," in *Robotics: Science and Systems (RSS)*, Jun. 2018. 2.4

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Intl. Conf. on Computer Vision (ICCV)*, Nov. 2011, pp. 2564–2571. 2.4, 3.1

[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, pp. 91–110, Nov. 2004. 3.1

[32] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014 3.1

[33] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692 3.1

[34] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003. 3.1

[35] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975. [Online]. Available: http://doi.acm.org/10.1145/361002.361007 3.2.1

[36] P. J. Burt and E. H. Adelson, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, M. A. Fischler and O. Firschein, Eds. Morgan Kaufmann Publishers Inc., 1987. [Online]. Available: http://dl.acm.org/citation.cfm?id=33517.33571 3.2.3

[37] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2013, pp. 5724–5731. 4.1.1

[38] M. Žefran and V. Kumar, "Interpolation schemes for rigid body motions," *Computer-Aided Design*, vol. 30, no. 3, pp. 179–189, Mar. 1998. 4.1.3

[39] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society America A*, vol. 4, pp. 629–642, 1987. 4.1.4

[40] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012. 4.2, 4.3.1, 5.3.2

[41] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," *Georgia Tech Technical Report*, 2012. 4.3.1, 5.4.1

[42] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2014, pp. 1524–1531. 4.3.2

[43] "Cloudcompare user manual," https://www.danielgm.net/cc/doc/qCC/CloudCompare%20v2.6.1%20-%20User%20manual.pdf. 4.3.2

[44] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 573–580. 4.3.2, 4.3.3, 4.6b

[45] O. Wasenmüller, M. Meyer, and D. Stricker, "CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2," in *Winter Conf. on Application of Computer Vision (WACV)*, Mar. 2016, pp. 1–7. 4.3.3

[46] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, pp. 571–583, May 2015. 5.3, 5.3.1, 5.3.1

[47] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," in *SIGGRAPH*. New York, NY, USA: ACM, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1275808.1276478 5.3.5, 5.3.5

[48] J. Park, Q. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Intl. Conf. on Computer Vision (ICCV)*, Oct. 2017, pp. 143–152. 5.4.3, 5.12