ENHANCED 3D CAPTURE FOR ROOM-SIZED DYNAMIC SCENES WITH COMMODITY DEPTH CAMERAS

Mingsong Dou

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2015

Approved by:

Henry Fuchs

Tat-Jen Cham

Enrique Dunn

Jan-Michael Frahm

Shahram Izadi

Anselmo Lastra

ABSTRACT

MINGSONG DOU: ENHANCED 3D CAPTURE FOR ROOM-SIZED DYNAMIC SCENES WITH COMMODITY DEPTH CAMERAS.
(Under the direction of Henry Fuchs .)

3D reconstruction of dynamic scenes can find many applications in areas such as virtual/augmented reality, 3D telepresence and 3D animation, while it is challenging to achieve a complete and high quality reconstruction due to the sensor noise and occlusions in the scene. This dissertation demonstrates our efforts toward building a 3D capture system for room-sized dynamic environments. A key observation is that reconstruction insufficiency (*e.g.*, incompleteness and noise) can be mitigated by accumulating data from multiple frames. In dynamic environments, dropouts in 3D reconstruction generally do not consistently appear in the same locations. Thus, accumulation of the captured 3D data over time can fill in the missing fragments. Reconstruction noise is reduced as well.

The first piece of the system builds 3D models for room-scale static scenes with one hand-held depth sensor, where we use plane features, in addition to image salient points, for robust pairwise matching and bundle adjustment over the whole data sequence.

In the second piece of the system, we designed a robust non-rigid matching algorithm that considers both dense point alignment and color similarity, so that the data sequence for a continuously deforming object captured by multiple depth sensors can be aligned together and fused into a high quality 3D model. We further extend this work for deformable object scanning with a single depth sensor. To deal with the drift problem, we designed a dense

nonrigid bundle adjustment algorithm to simultaneously optimize for the final mesh and the deformation parameters of every frame.

Finally, we integrate static scanning and nonrigid matching into a reconstruction system for room-sized dynamic environments, where we prescan the static parts of the scene and perform data accumulation for dynamic parts. Both rigid and nonrigid motions of objects are tracked in a unified framework, and close contacts between objects are also handled.

The dissertation demonstrates significant improvements for dense reconstruction over state-of-the-art. Our plane-based scanning system for indoor environments delivers reliable reconstruction for challenging situations, such as lack of both visual and geometrical salient features. Our nonrigid alignment algorithm enables data fusion for deforming objects and thus achieves dramatically enhanced reconstruction. Our novel bundle adjustment algorithm handles dense input partial scans with nonrigid motion and outputs dense reconstruction with comparably high quality as the static scanning algorithm (*e.g.*, KinectFusion). Finally, we demonstrate enhanced reconstruction results for room-sized dynamic environments by integrating the above techniques, which significantly advances state-of-the-art.

TABLE OF CONTENTS

LIST OF FIGURES

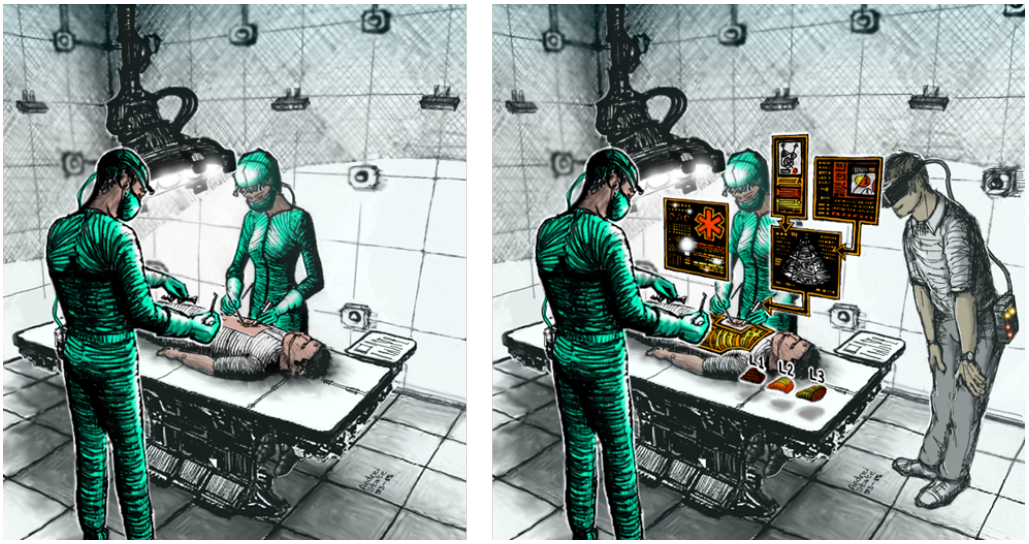
LIST OF TABLES

CHAPTER 1: Introduction

Reconstructing a scene digitally, or capturing its 3D geometric model, has been one of the most challenging and fundamental problems in Computer Vision since its emergence, and high quality 3D reconstruction is essential for many potential but yet unrealized applications in virtual and augmented reality. For example, virtual reality applications, such as surgical training programs and science demonstration, could continually record raw imagery (and sound) inside operating rooms; when noteworthy events occur (*e.g.*, a difficult trauma surgery case), reconstruction and subsequent annotation would be triggered, to generate an immersive virtual presence training module. To be effective, such modules should encourage free navigation and user-selectable viewpoints and also have high reconstruction fidelity without any undesired artifacts. Figure 6.1 illustrates the above concept.

3D reconstruction is also an important aspect for 3D telepresence, where users get custom perspectives of the remote scene (or a combination of the local and remote scene) through a 3D display wall or head-mounted displays. Figure 1.2 illustrates the concept of a three-way 3D telepresence system for informal gathering. Different from traditional videoconferencing systems such as CISCO TelePresence IX5000 series[1], users are not restricted to sit at specific positions and are able to engage communications with remote users as they do with local ones. We attempted to achieve such a system with a 2D image synthesis approach in our early work (Dou et al., 2012). Although we solved some aspects of the problem (*e.g.*, eye-contact for

[1]http://www.cisco.com/c/en/us/products/collaboration-endpoints/ix5000-series/index.html



(A) 3D capture during medical procedure (B) immersive experience of the 3D reconstruction

Figure 1.1: 3D Medical environment reconstruction for medical training. (A) Operating room is instrumented with various capture devices such as depth and pan-tilt-zoom cameras; medical personnel wear miniature cameras. (B) The procedure has been reconstructed in three dimensions and annotated (orange). A medical student (grey) experiences the reconstructed sequence immersively within a modern low-latency, wide-field-of-view head-worn display, navigating in space and time.

users right in front of the display), lack of 3D information of the scene prevents us from generating proper views for all users. This early attempt motivated us to work on 3D room-sized reconstruction.

The above applications demand sufficiently accurate 3D reconstruction to enable unencumbered and seamless immersion into the reconstructed environment. In addition, most activities of interest involve not just static scenes, but also moving objects and people which are much more difficult to reconstruct. Thus, in this dissertation we focus our research on a dynamic 3D-plus-time reconstruction of an actively evolving scene in indoor environment.

Combined with the recent emergence of high-quality, low-latency head-mounted displays (*e.g.*, Oculus Rift, Sony Morpheus, and Microsoft HoloLens), we can imagine that 3D dynamic

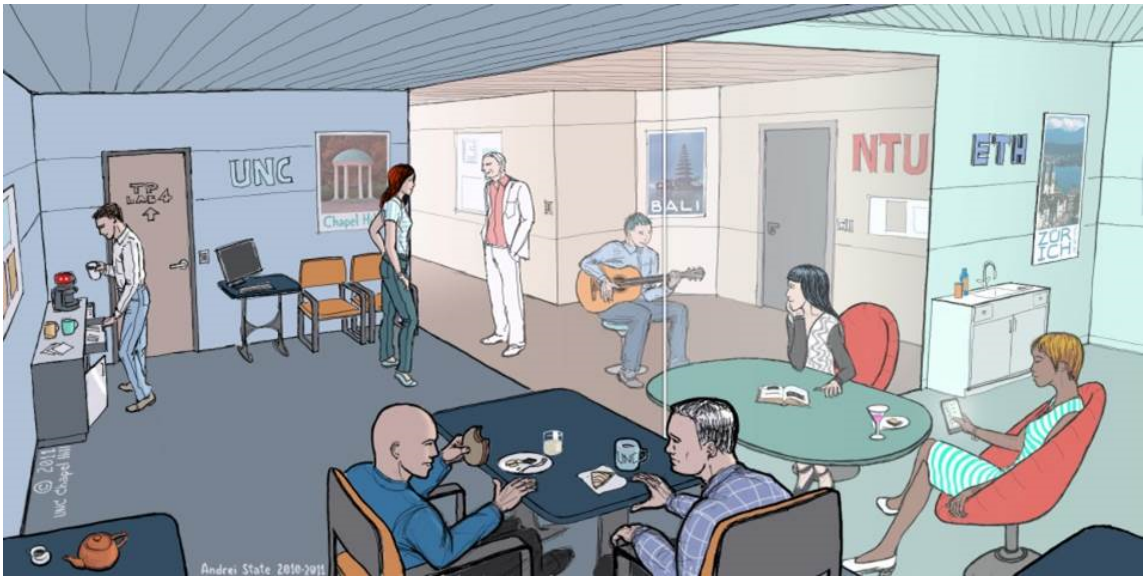


Figure 1.2: Illustration of concept of 3D telepresence. Every location is reconstructed in 3D and every person gets a custom view of the remote scenes from the 3D display. The wall on the right side is instrumented with autostereoscopic displays, through which people in the local room see the combination of two remote rooms.

scene reconstruction can impact society profoundly. It could influence the way we share information. Instead of sharing photos on social media, people could share recorded 3D videos that can be immersively experienced. It could also change the way we interact with other people and the environment, via applications such as immersive virtual visits to interesting locations and virtual attendance at events such as concerts or a class.

Today, the possibility of accurate high-quality 3D reconstruction (especially for indoor environments) gets ever more promising due to the advances in camera technology, especially the introduction of depth sensors (*e.g.*, Microsoft Kinect and Intel RealSense). These depth sensors are affordable consumer-grade models, opening the possibility of populating a relevant location with high sensor density, and making deployment of such 3D capture technology affordable in many locations and situations. In this dissertation, we take advantage of these low-cost depth

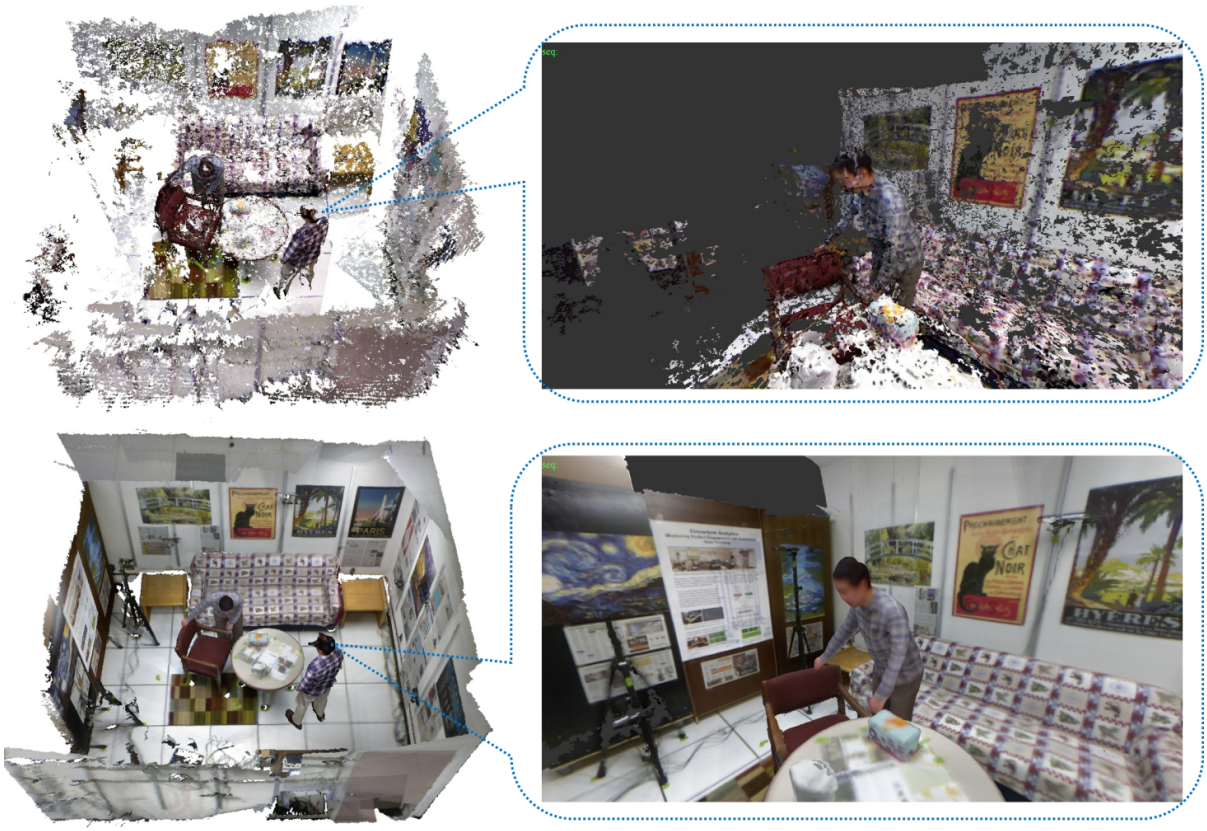


Figure 1.3: Room-sized reconstruction for a dynamic scene by performing data accumulation and background prescan. Top: input depth points from 10 Kinects; Bottom: enhanced reconstruction. The rendering from the perspective of the man standing at right is shown on the right. Note the low quality on the input data despite there being 10 cameras. Note also we did not shake the sensors to reduce some level of sensor interference (Maimone and Fuchs, 2012; Butler et al., 2012).

sensors (*i.e.*, Kinect) to achieve high quality reconstruction for room-sized dynamic scenes.

1.1 Challenges and Opportunities

To approach the possibilities described above, simply populating the walls of the space to be captured with many depth sensors does not yield the required high-quality reconstructions (as shown in Figure 1.3). One major artifact is the incompleteness of the reconstructed models. Such incompleteness can be caused by dropouts in the depth sensor data. For example, the

Figure 1.4: Data accumulation for a person turning around in front of depth sensors. Surfaces after fusing 1, 5, 50, and 150 frames of data.

Kinect sensor has difficulty reconstructing low or high reflective surfaces. In addition, incompleteness is also caused by occlusion, which includes self-occlusion as well as inter-occlusion of objects. Yet another undesired artifact comes from the significant amount of noise in the data from depth sensors (Maimone and Fuchs, 2011; Beck et al., 2013).

Fortunately, the dynamic properties of the scene raise opportunities for improved dense reconstruction. One observation is that reconstruction insufficiency (*i.e.*, incompleteness and noise) can be mitigated by accumulating data from multiple frames (*e.g.*, KinectFusion (Newcombe et al., 2011) for static scene reconstruction). In dynamic environments, dropouts in 3D reconstruction generally do not consistently appear in the same locations. Thus, accumulation of the captured 3D data over time can fill in the missing fragments. Reconstruction noise will be reduced as well. Figure 1.4 shows the improvement on the reconstruction by accumulation more and more data frames. However, before consecutively captured data frames can

be accumulated, they must be aligned spatially, which requires tracking corresponding surface fragments accurately over time. An insufficient alignment will lead to smoothing of surface details and geometric features. To use this data accumulation concept in real-world applications, we must deal with the following challenges:

- **Nonrigid movement.** In many cases, especially when processing deformable geometry such as human bodies, a nonrigid alignment algorithm must be employed. An ideal alignment algorithm should be robust to data dropouts on surfaces and work for general surfaces (*i.e.*, without assumptions on the surface shape).

- **Dynamic topology.** In real scenarios, there are many topology changes in the geometry of the scene caused by people and objects engaging contact (e.g., shaking hands, sitting on a chair).

- **Drift.** Even for rigid alignment of non-deformable objects, drift occurs when accumulating a sequence of geometric data over a period of time and thus over a number of capture events (frames). As alignment error accumulates, scanned surfaces do not close seamlessly.

1.2 Thesis approach and overview

In this dissertation, we developed several systems that employ the data accumulation strategy to enhance surface reconstruction for indoor dynamic environment while tackling aforementioned challenges. Specifically,

- We perform pre-scanning for static part of the environment by rigidly aligning and fusing

data from one single hand-held depth sensor.

- We also perform data accumulation for the dynamic part of the scene via nonrigidly aligning data to the same reference pose. The data either come from the depth sensors mounted over the walls or from a single depth sensor (hand-held or fixed). Deforming the accumulated model to align the data at every frame results with enhanced dense reconstruction.

- We combine static pre-scanning with nonrigid accumulation, leading to a dense reconstruction system for a room-sized dynamic scene. In addition, we estimate the rigid movements for static objects (*e.g.*, chairs, tables) and handle close contact between objects.

We will introduce the static scanning in Chapter 2, the dynamic object scanning with multiple depth sensors in Chapter 4, the nonrigid surface reconstruction with one single depth sensor in Chapter 5, and the combination of static pre-scanning and nonrigid accumulation for room-sized dynamic reconstruction in Chapter 6. We will conclude the dissertation in Chapter 7 with summarizing our contribution and listing future work.

To deal with the challenges in real world situations, we developed a nonrigid alignment algorithm that robustly aligns data of deforming objects pair-wisely, by taking into account the measurement of both dense surface alignment and color consistency. We do not make any shape assumption for objects, and thus the algorithm works for general surfaces (up to a fixed topology). It is a partial-to-partial alignment algorithm and allows for decent amount of missing data on surfaces. Our nonrigid alignment algorithm is introduced in Chapter 3.

Drift is a ubiquitous problem when aligning a sequence of data together, happening in both rigid and nonrigid cases. For static scanning, we incorporate plane features into the bundle adjustment framework to mitigate the error over the whole data sequence, in observance of planes being the dominant feature for the indoor environment. For dynamic surface reconstruction from a single depth camera, we developed a dense nonrigid bundle adjustment technique which simultaneously optimize both the dense geometry and deformation parameter. The new bundle adjustment technique takes dense partial structure as input and outputs the optimal fused surface and nonrigid motion. For the data accumulation from multiple units of depth sensors, since the data is relative complete at a frame (compared with the case of a single depth sensor), we take a computationally less intense algorithm. We estimate the deformation of the currently accumulated model to the current data observation and build the point-to-point correspondence, from which a backward deformation that deforms data to the model is estimated. Then we fuse the current data into the model and repeat the above procedures. This strategy would be proven effective in Chapter 4.

Surface topology change is also a ubiquitous problem and is very difficult to solve. Throughout the dissertation, we make various assumptions to overcome the problem. In Chapter 4, we divide the dynamic object reconstruction system into two stages: scanning and tracking. During scanning state, we restrict the user to maintain roughly a 'T'-pose and later lift such a constraint during tracking stage. In Chapter 5, we assume the topology is consistent throughout the whole data sequence. In Chapter 6, we partition the scene into different categories, and perform segmentation on the pre-scanned static scene. Building a system without any topology restriction would be future work.

Note that we focus on reconstruction quality rather than on real-time processing in this dissertation, *i.e.*, the data are captured in real time while being processed offline. Such techniques are still valuable as many potential applications do not require real-time reconstruction but have stringent reconstruction quality requirement. It is the case with the aforementioned medical and surgical training, where high quality reconstruction and faithful presentation of delicate and intricate 3D manipulations is paramount. Furthermore, with continuous advances in computing power and GPU-enabled parallelization, it can be reasonably expected that the proposed techniques will be applied to real-time situations in the near future, allowing telepresence-type interactions.

1.3 Related work

We will give a brief summary of work on 3D reconstruction here. More related work for each system component is provided at following chapters. Traditional 3D reconstruction algorithms, including stereo vision (Scharstein and Szeliski, 2002) and multi-view vision (Seitz et al., 2006) algorithms, rely on color cameras and have difficulties when processing textureless surfaces. Recent consumer-targeted depth sensors, such as the Microsoft Kinect and Intel 3D perception sensor, dramatically changed the field due to their high performance-price ratio. These sensors either employ the infrared projector and camera system or time-of-flight technique and are able to give real-time reliable depth maps even for textureless areas. These lead to static reconstruction approaches such as KinectFusion (Newcombe et al., 2011; Izadi et al., 2011), commercial softwares such as ReconstructMe[2] and Matterport[3].

[2]http://reconstructme.com
[3]http://matterport.com

The work of using Kinects for realtime reconstruction application with light-weight processing on the depth maps include (Beck et al., 2013; Kuster et al., 2011), where the reconstruction is far from complete and often has a high level of noise. Other reconstruction systems that require highly controlled environment or employ expensive hardware include USC Light Stage (Vlasic et al., 2009; Ghosh et al., 2011) and ETH Blue-C project (Gross et al., 2003).

Instead of the reconstruction of static scenes, work exists on building one model from a data sequence of a deforming object (Tong et al., 2012; Hirshberg et al., 2012; Weiss et al., 2011; Li et al., 2013; Zeng et al., 2013; Liao et al., 2009), but these systems either does not deal with dramatic surface deformation or rely heavily on shape prior knowledge. In addition, Motion Capture systems (De Aguiar et al., 2008; Gall et al., 2009; Ballan and Cortelazzo, 2008; Vlasic et al., 2008; Wu et al., 2013; Ye et al., 2012; Starck and Hilton, 2007; de Aguiar et al., 2007) deliver reliable reconstruction of the human body from a sequence of input color and/or depth feeds. But these systems require a prescanned model or template and complicated skinning and rigging preprocessing, which prevents their applications for reconstructing general shapes.

CHAPTER 2: Static Scene Reconstruction

In this chapter, we introduce our 3D scanning system for room-sized static scenes using one depth sensor/RGB-D camera (*e.g.*, Microsoft Kinect). RGB-D cameras are new addition to vision sensors for 3D reconstruction, which output RGB images and corresponding depth images at video frame rate. Recent approaches demonstrated the use of a single hand-held RGB-D camera for scanning an indoor environment (Henry et al., 2010; Izadi et al., 2011; Newcombe et al., 2011; Neumann et al., 2011; Lieberknecht et al., 2011) and even large scenes (Chen et al., 2013; Nießner et al., 2013; Zhou et al., 2013).

These algorithms roughly fall into two major categories: (1) point cloud based registration such as Iterative Closest Point (ICP) and (2) visual feature based Structure from Motion (SfM). KinectFusion system (Izadi et al., 2011; Newcombe et al., 2011) utilizes ICP to align one frame's structure with previously captured data. While achieving subpixel accuracy in certain environment settings (*e.g.*, table-sized scene), the major drawback is that ICP relies only on distinctive geometric information and would be confused when scanning geometrically smooth region (*e.g.*, a wall), leading to *drift*.

Structure from Motion (SfM) is another classic technique for 3D reconstruction. Standard SfM algorithms only use image feature points, so it fails when very few matched features are found or extracted features are not well distributed over the scene during pairwise matching stage. As demonstrated later, in the worst case–when the whole room contains very few visually salient features—no frame will be registered together with SfM. Therefore, ICP and SfM

algorithms are expected to fail in challenging cases where a scene has very few image and geometry salient features.

However, we notice that additional high level contraints exist in the depth channel. As shown in Fig. 2.4, although there is no image feature correspondence detected around the ceiling, the two frames can still be well aligned by extracting planes from the depth map and searching for the correct plane matches. By combining high-level depth information and low-level image salient features, our system handles well previously mentioned challenging situations, which cannot be solved alone by ICP or SfM.

In the SfM literature, the final step is usually global error mitigation through bundle adjustment (BA) (Snavely et al., 2007, 2006; Crandall et al., 2011). This global adjustment step is crucial for building a model for a large scene such as an entire room. However, as shown in Chapter 5, it takes huge efforts (*e.g.*, computation power) to directly incorporate dense point cloud into the BA framework due to implicit point association. Instead, this chapter proposes to use a compact planar representation of dense point clouds and integrate it into the traditional BA formulation.

To summarize, the major contributions of the indoor scanning system introduced in this chapter include:

- First, a robust pair-wise matching algorithm across frames via matching both extracted planes and RGB image visual features. Our evaluation demonstrates that planes better constrain the reconstruction problem in the aforementioned challenging cases (*i.e.* low-texture low geometry information regions).

Figure 2.1: Results of the scanning system on datasets (a)Rm. FB220, (b)Rm. SN277 and (c)Rm. SN353. Accumulated point clouds (colors are added to distinguish points from different planes), zoomed view, and planes resulted from our algorithm are shown from left to right.

- Second, a novel formulation for incorporating plane correspondences (in addition to visual feature correspondence) into the bundle adjustment framework. Planes are more compact than points and have clear associations across frames. This bundle adjustment step eliminates the drift problem that hurts KinectFusion.

- In addition, our proposed algorithm results in a piecewise planar representation for planar parts of the scene. Such compact representation of the scene can be deployed for applications such as noise reduction, data compression, segmentation, etc.

While any primitive representation can be used with our method, a piecewise planar represen-

tation is most natural for man-made indoor environments, due to the dominant existence of planar surfaces, such as walls, floors, etc.

We evaluate our approach on several real world indoor datasets. Some have a lot of textureless regions, which confuse SfM algorithms; some also have areas with significant geometrical ambiguity, such as large pieces of walls, which confuse ICP algorithms. By combining low level appearance features and high level geometric primitives, our algorithm handles these challenges well and significantly improves the reconstruction results.

2.1 Other Related Work

Plane constraints have been proved useful in other works for 3D reconstruction or 3D mapping. Sinha et al. (2009) and Furukawa et al. (2009) recovered a piecewise planar representation of the scene during the stereo procedure; Gallup and Frahm (2010) detected the planar surface based on the trained information to improve the results of stereo for urban scene reconstruction. Lee et al. (2011) used the constraints from coplanar feature points for visual SLAM. Different from these works, we use planes extracted from relatively accurate depth maps, which serve as an independent piece of information from features in RGB images. Pathak et al. (2010) and Pathak et al. (2009) also extracted planes from a depth camera and performed plane matching to register frames. However, without the help of salient feature points, planes alone can not determine transformations between frames that contain only simple geometries such as walls. Additionally, they did not exploit the plane alignment constraint for the global error mitigation.

Our work also relates to the research on SfM and Bundle Adjustment (Pollefeys et al., 2004; Snavely et al., 2006; Crandall et al., 2011; Steffen et al., 2010). Again, the novelty of

our method is that we perform bundle adjustment on both image features and planes.

2.2 System Overview

To build a complete room model, a sequence of RGB-D images is captured from a hand-held depth sensor. The camera intrinsics are pre-calibrated, thus we only need to estimate the camera poses (extrinsics, namely camera rotations and translations) to reconstruct the room. Note that depth calibration (detailed in Chapter 6) helps the reconstruction quality for large scene.

For every frame in the sequence, we extract SIFT features from RGB color image and planar patches from the depth maps (detailed in Section 2.3). The corresponding 3D coordinate for each image feature point is calculated under its camera coordinate system, given that the depth value at its image location is known. We call these 3D points originating from image features simply as "features" or "points" in order to differentiate them from planes extracted from depth maps. We remove features that are located at discontinuity regions in the depth map as their depth values are typically inaccurate.

Next, we perform a pairwise matching over the data sequence to find the matched frames. Robust pairwise matching results are achieved by using both planar surfaces and features. For each matched frame pair, the matched features and planes are found, together with their underlying geometric transformation. We will introduce the RANSAC-based matching algorithm that handles both points and planes matches in Section 2.4. To save computation time, We do not test every frame pair for match exhaustively in the whole data sequence. Instead, similar to (Henry et al., 2010), the time coherence of the sequence is explored and some key frames

are automatically selected during the matching procedure. When a new frame comes, it is only matched against previous key frames and a few neighboring frames.

Finally, we run a bundle adjustment algorithm on all the matched features and planes from all pairs of the matched frames to optimize camera poses globally. The bundle adjustment algorithm is carried out directly in 3D space instead of 2D image space thanks to the 3D location information computed from the depth channel. Different from traditional bundle adjustment, planes provide strong constraints for camera pose estimation in indoor environment. This extended bundle adjustment algorithm is introduced in Section 5.6. We evaluate the proposed system with plane constraints in Section 6.4 qualitatively and quantitatively and demonstrate significant improvement over state-of-the-art reconstruction algorithms in real world indoor datasets.

2.3 Planar Surface Extraction from Depth Map

Given the camera intrinsics, a depth map can be transformed to a point cloud, and vice versa. Therefore we use these two terms interchangeably. In the literature of plane extraction from point cloud, region growing (Poppinga et al., 2008) and voting (Borrmann et al., 2011) are two popular techniques. Region growing algorithms scatter some initial seeds over the image, and gradually merge neighboring pixels belonging to the same plane. The voting algorithms transform all points into plane parameter space. The peaks in the parameter space correspond to the consensus planes in the original space. A plane in 3D space is represented as

$$\mathbf{n}^T\mathbf{x} - d = 0,$$

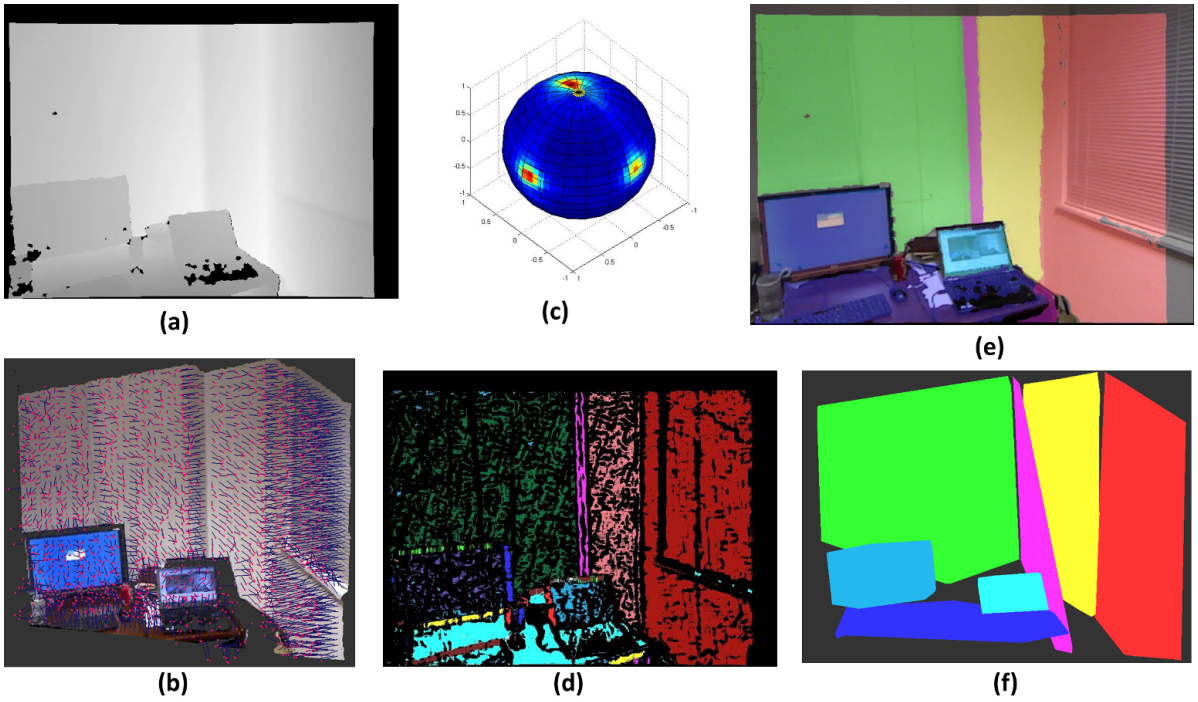


Figure 2.2: Planar surface extraction via voting in plane parameter space. (a) input depth map; (b) normal vectors of local planes at each pixel; (c) votes in the plane parameter space; (d) pixels voting for the same peaks; (e) final plane labels; (c) the plane segments in 3D

where $\mathbf{n} = \{n_x, n_y, n_z\}$ is the plane normal—a 3D unit vector; d is the distance of the plane to the camera optical center. We force the plane normal point away from the origin, resulting a positive d. From now on, we let $\mathcal{P} = \langle \mathbf{n}, d \rangle$, representing a plane under its individual camera coordinate system for that frame.

Here a plane voting algorithm is used. Local planes are first fitted for every pixel in the depth map using its neighboring pixels (as shown in Figure 2.2(b)). Then these local planes vote in the plane parameter space to find consensus planes (Figure 2.2(c)). Spherical coordinate

Figure 2.3: Example of extracted planes.

$\langle \rho, \theta, \phi \rangle$ is used for plane voting, *i.e.*,

$$\begin{aligned} \theta &= \cos^{-1} n_z \\ \phi &= \tan^{-1} \frac{n_y}{n_x} \\ \rho &= d, \end{aligned} \tag{2.1}$$

After finding the peaks in the plane parameter space, the plane parameters are re-estimated by fitting planes from all the pixels voting for the same peak (Figure 2.2(d)). Finally, each pixel is assigned either to one of the detected planes or as non-plane if the distance to all planes is too large (Figure 2.2(e)). Plane parameters are refined again from their associated pixels, and the convex hull $\{\mathbf{v}_i\}_{i=1}^{K}$ of a plane segment is found to indicate its boundary (Figure 2.2(f)). More

examples of extracted planes are shown in Figure. 2.3.

2.4 Robust Pair-wise Matching

Given the detected planes and features for two frames, next we find the matched features and matched planes if any. The geometric transformation that aligns these two frames is also estimated accordingly. Conventionally, we represent the transformation as a rotation matrix R and a translation vector T, satisfying

$$X^r = RX^l + T,$$

where X^l and X^r are 3D points under two camera coordinate systems respectively (left and right). To reduce the searching space of matched planes and features, the initial feature and plane match set are found before being refined with RANSAC.

2.4.1 Plane Matching Hypothesis

While the initial feature match set can be extracted efficiently by checking the similarity of SIFT descriptors, it is more difficult to find an initial plane match set. Appearance-only matching algorithm is not reliable enough to find plane correspondence, as two frames can observe disjoint parts of a same plane which visually can be dramatically different. Here we use the plane relative angle as the additional clue. Within an authentic plane match set (two groups of planes, one for each frame), relative angles between planes are invariant even with different viewing perspective.

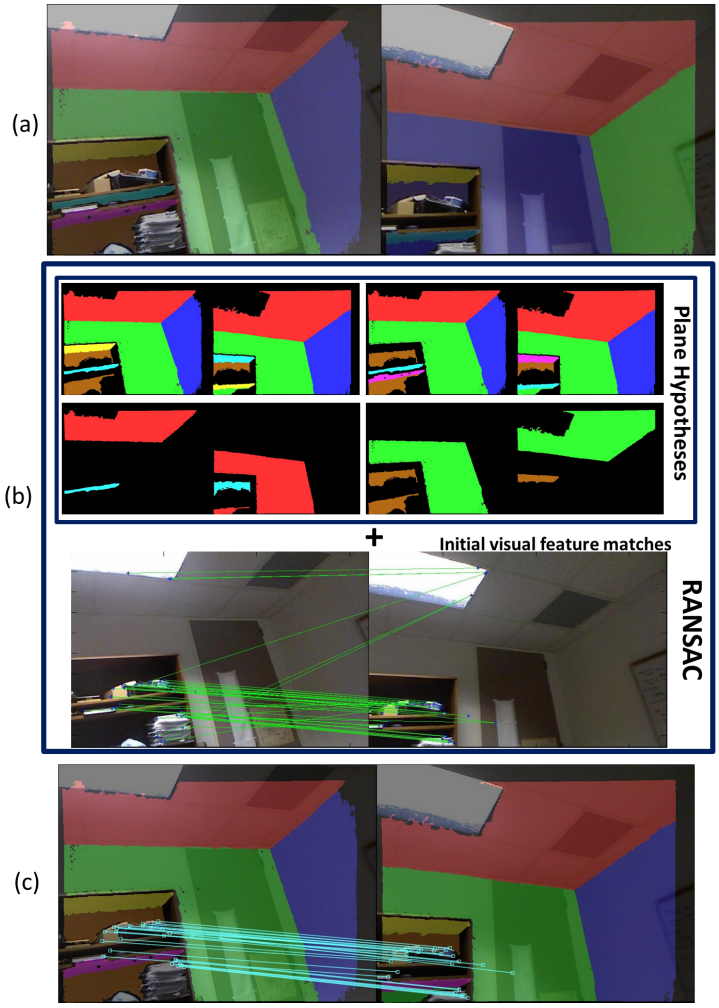


Figure 2.4: Robust Pairwise Matching of Both Planes and Features. (a) two input frames and detected planes. (b) four plane matching hypotheses are shown on top (same color indicates a plane match), and the initial feature matches are shown on bottom. (c) matched features and planes after RANSANC.

Generally, combining the above relative angle and appearance constraint does not achieve one unique plane correspondence. For example, when three planes are mutually perpendicular and have similar appearance, there are totally six different plane matching hypotheses. We call the plane correspondence for two frames (containing a subset of planes in one frame and the subset of the matching planes in the other frame) as a **plane matching hypothesis**. The algorithm introduced in the remaining section reaches multiple hypotheses, which is fine since

the following RANSAC stage is able to find a unique solution.

Finding two matched plane subsets is equivalent to finding a rotation matrix that rotates the plane normals in one set to match the plane normals in the other set. Theoretically, a brute force search on the rotation matrix space is a possible solution. That is, we apply all possible rotation matrices on the planes of one frame, and find a subset of planes in this frame that have corresponding planes with similar plane normals and appearance in the other frame.

However, the above brute force algorithm is not practical as it searches a 3D space for the rotation matrix. Similar to (Pathak et al., 2009), we take advantage of the fact that a rotation matrix can be decomposed into a rotation axis and a rotation angle about the axis. Specifically, we repeat the following procedures:

1. Pick one pair of planes $\langle \mathcal{P}_i^l, \mathcal{P}_{i'}^r \rangle$ that have similar appearance from two frames, and find a rotation matrix R that rotates normal vector $\mathbf{n}_i$ to $\mathbf{n}_{i'}$, i.e., $R\mathbf{n}_i = \mathbf{n}_{i'}$.

2. Apply R on all the planes in the first frame. By now, $\mathcal{P}_i^l$ and $\mathcal{P}_{i'}^r$'s normals are aligned, but most likely not any other planes in the two views.

3. Rotate the planes of the first frame about axis $R\mathbf{n}_i$ to find their matched planes with similar plane normals and appearance in the second frame. In this way, we search along only one dimension—the rotation angle about axis $R\mathbf{n}_i$.

Note that using d in plane parameter as an extra constraint will give better plane matching hypotheses but also increase the search space (6 DOF).

For each plane segment, we calculate a hue-saturation joint histogram $\mathbf{h}^{HS}$ to represent its color information and an intensity histogram $\mathbf{h}^{I}$ to represent its texture information. The

appearance similarity of the two planes is defined as the overlapping area of their histograms, i.e.,

$$\sum_i \min(\mathbf{h}_1^{HS}(i), \mathbf{h}_2^{HS}(i)) + \sum_i \min(\mathbf{h}_1^{I}(i), \mathbf{h}_2^{I}(i)).$$

2.4.2 RANSAC on one plane matching hypothesis and the feature match set

The initial feature match set might contain spurious pairs, as might a plane matching hypothesis. We use RANSAC to find the underlying geometric transformation between two frames and refine the match sets. When a feature or plane pair fits a candidate transformation, it supports this transformation (details are given later). RANSAC returns a transformation that collects most supports from match sets. Note RANSAC is extended to take both the feature match set and plane match set as input. Since Section 2.4.1 might return multiple plane matching hypotheses, we run RANSAC multiple times on every plane matching hypothesis together with initial feature match set. Within multiple transformations returned by RANSAC, again we pick the one that collects most supports.

Randomly sample matched pairs

RANSAC randomly samples a minimum number of matched pairs from initial match set to determine a rigid transformation candidate. In our case, there are two categories of matched pairs–matched features and matched planes. As elaborated in (Pathak et al., 2010), three planes with linearly independent normal vectors uniquely determine a transformation; similarly three non-collinear features uniquely determine a transformation. Additionally, two nonparallel planes and a feature also determine a transformation, as do two features and a plane with a

different normal direction than the vector connecting the two feature points. Our randomly sampled match subsets are evenly distributed over the above four cases—three planes, three features, two planes with one feature, and two features with one plane.

Calculating the transformation from pairs of matches

Inside each loop of RANSAC, the transformation needs to be calculated from the randomly sampled matches (the seeds). We consider a general case, where there are n pairs of matched planes $\mathcal{S} = \{\langle \mathcal{P}_i^l, \mathcal{P}_{i'}^r \rangle\}$, and m pairs of matched features $\mathcal{T} = \{\langle \mathbf{f}_i^l, \mathbf{f}_{i'}^r \rangle\}$. The upper-index l and r indicates which frame the data belong to, and we will ignore them whenever it does not cause confusions. A transformation $\langle R, T \rangle$ is estimated to minimize the overall distance between matched items. That is,

$$\min \sum_{\langle \mathcal{P}_i^l, \mathcal{P}_{i'}^r \rangle \in \mathcal{S}} \mathbf{D}_{pln}^2(Q(R, T, \mathcal{P}_i^l), \mathcal{P}_{i'}^r) \; + \sum_{\langle \mathbf{f}_i^l, \mathbf{f}_{i'}^r \rangle \in \mathcal{T}} \mathbf{D}_{pt}^2(R\mathbf{f}_i^l + T, \mathbf{f}_{i'}^r). \tag{2.2}$$

where $Q(\cdot)$ applies transformation $\langle R, T \rangle$ to $\mathcal{P}_i^l$ and is detailed later, while $\mathbf{D}_{pln}(\cdot)$ and $\mathbf{D}_{pt}(\cdot)$ are distance functions for planes and features respectively. The distance between features is simply defined as Euclidean distance. However, the Euclidean distance of plane parameters $\langle \mathbf{n}, d \rangle$ is not a good measurement of the plane distance as shown in Fig. 2.5, mainly because the planes in our case actually mean "plane segments" which have boundaries.

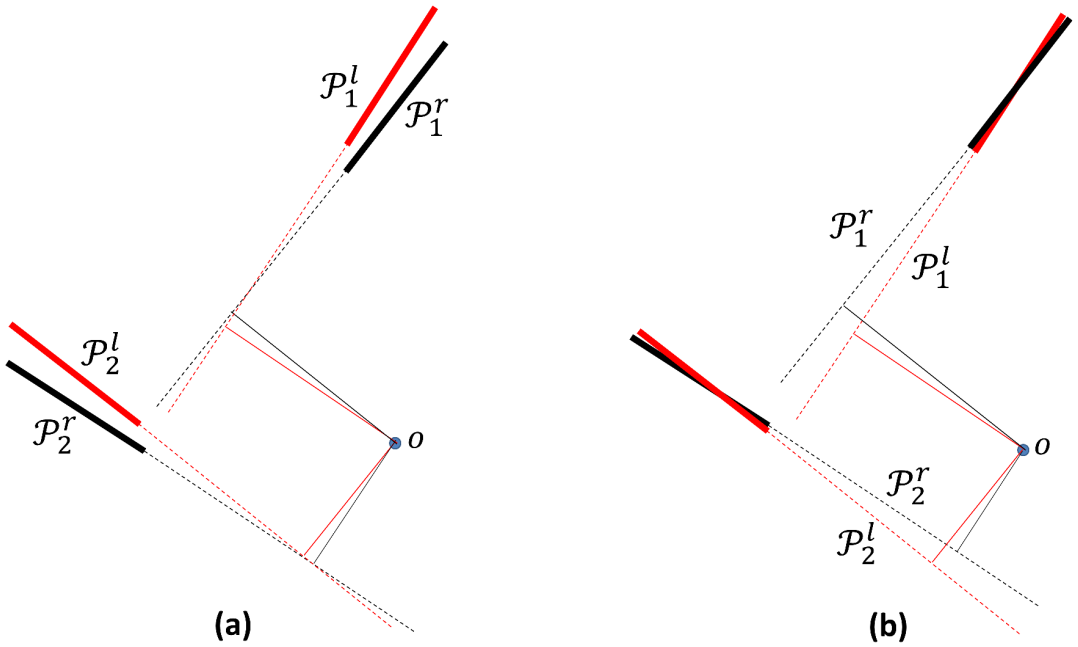


Figure 2.5: The closeness on planes parameters shown in (a) does not equal to the closeness of plane segments shown in (b). The solid line segments denote the plane segments, and o is the origin of the world coordinate. In (b), even though each pair of matched planes have much larger difference on the origin-to-plane distance d, they are better aligned comparing with (a).

Distance between plane segments

Instead of measuring Euclidean distance between plane parameters, we measure the distances from the boundary points (convex hull) of one plane segment to its matched plane. Ideally, we could use the sum of the point-to-plane distances from all points in one plane segment to the other plane, since these points are the direct observations of a plane. However it is computationally more expensive given the number of points in a plane segment. It can be proven the convex-hull-to-plane distance is the upper-bound of the sum of point-to-plane distances[1].

[1]Each observed point p assigned to a plane could be represented by a linear combination of the convex hull vertices $\{v_i\}$, i.e., $p = \sum_i c_i v_i + \epsilon$; where $\sum_i c_i = 1$ and ϵ accounts for the fact that the point p might not lie exactly on the fitted plane. Since the point-to-plane distance is a convex function, the weighted sum of convex-hull-vertex-to-plane distances is statistically the upper-bound of the distance from an observed point to the world plane. Taking all the points associated with one plane into consideration, the sum of squared point-to-plane distances (upper-bound) is $c \sum_i w_i \mathcal{D}^2(v_i)$, where c is the number of the points; $\mathcal{D}(\cdot)$ measures point-to-plane distance;

Let's assume we are measuring the distance between two plane segments $\{\mathcal{P}_i^l, \mathcal{P}_{i'}^r\}$ under the transformation $\langle R, T\rangle$. The the convex hull vertices of two plane segments are $\{\mathbf{v}_{i,k}^l\}_{k=1}^{K_1}$ and $\{\mathbf{v}_{i',k}^r\}_{k=1}^{K_2}$ respectively. As shown in (Pathak et al., 2010), after applying the transformation $\langle R, T\rangle$ to plane $\mathcal{P}_i^l$, its plane parameter becomes $\langle R\mathbf{n}_i,\ \ d_i + \mathbf{n}_i^T R^T T\rangle$, and its convex hull vertices becomes $\{R\mathbf{v}_{i,k} + T\}_{k=1}^{K_1}$. Then the distance between plane segments in Eq. 2.2 is defined as,

$$D_{pln}^2 = \sum_{i=1}^{K_1} w_{i,k} \|\mathbf{n}_{i'}^T R\mathbf{v}_{i,k} + \mathbf{n}_{i'}^T T - d_{i'}\|_2^2 + \sum_{i=1}^{K_2} w_{i',k} \|\mathbf{n}_i^T R^T \mathbf{v}_{i',k} - \mathbf{n}_i^T R^T T - d_i\|_2^2. \quad (2.3)$$

The first term in the right side of the equation measures the perpendicular distance from the transformed convex hull vertices on the first plane to the second plane, while the second term measures the distance between the vertices of the second plane and the transformed first plane. Each point-to-plane distance is given a weight $w_{i,k}$, satisfying $\sum_k w_{i,k} = 1$. Because vertices on the convex hull are not a uniform sample of the plane boundary, larger weights are given to vertices further away from their neighboring vertices.

Find the supporting pairs and choose the best transformation

For each transformation candidate, we count the support from the match sets. A pair of points will cast a constant vote value c_{pt} if their distance is closer than ϵ_{pt}, while a pair of planes cast vote value c_{pln} if their distance defined in Eq. 2.3 is less than ϵ_{pln} and two plane segments overlap after transforming one plane's segment to the other plane's image space. That is, only when a pair of planes are close enough and have overlap under one transformation, we

and w_i is the overall weight on a vertex satisfying $\sum w_i = 1$.

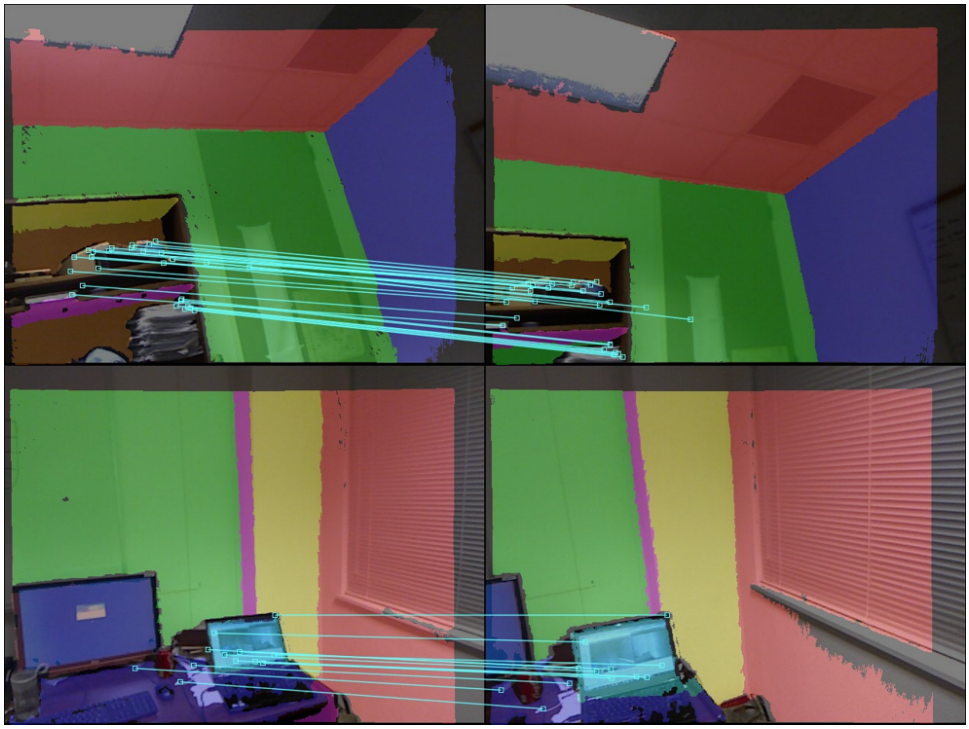


Figure 2.6: Two examples of point and plane match sets after refinement.

say they fit the transformation. In our experiment, plane pairs carry more weight than point pairs. Examples of the refined match set are shown in Figure 2.6.

2.5 Bundle Adjustment of Points and Planes

After performing the above robust pairwise matching algorithm over the whole RGB-D sequence, we have a large number of feature/plane match sets, and groups of matched features/planes are linked together according to . A set of linked features $\{\mathbf{f}_k^i\}_{i\in\mathcal{C}_k}$ is called a **feature track**, corresponding to the same 3D point $\mathbf{p}_k$ in a world coordinate system. The notation $\mathbf{f}_k^i$ represents a 3D feature point in the k-th visual feature track from the i-th frame, and $\mathcal{C}_k$ is the set of frame indices where all the indexed frames have feature corresponding to the 3D world point $\mathbf{p}_k$. Similarly from the plane matching sets, a number of **plane tracks** can be

found. Each plane track is composed of a set of linked planes $\{\mathcal{P}_j^i\}_{i\in\mathcal{D}_j}$ from various frames corresponding to the same world plane $\mathcal{Q}_j$. The notation $\mathcal{P}_j^i$ represents a plane in the j-th plane track and extracted from i-th frame, while $\mathcal{D}_j$ is the set of frame indices where all the indexed frames have the extracted planes corresponding to the world plane $\mathcal{Q}_j$. From now on, we always use i as the index of frames, j as the index of the plane track or its corresponding plane in the world space, and k as the index of the visual feature track or its corresponding point in the world space.

2.5.1 Problem Statement

Given M plane tracks $\{\{\mathcal{P}_j^i\}_{i\in\mathcal{D}_j}\}_{j=1}^M$, and K feature tracks $\{\{\mathbf{f}_k^i\}_{i\in\mathcal{C}_k}\}_{k=1}^K$, which are represented under their own camera spaces, the bundle adjustment problem is to simultaneously optimize the camera poses $\{R_i, T_i\}_{i=1}^N$ for N frames in the sequence, the plane parameters $\{\mathbf{n}_j, d_j\}_{j=1}^M$ for M planes $\{\mathcal{Q}_j\}$ in the world, and K point locations $\{\mathbf{p}_k\}_{k=1}^K$ in the world. As a usual practice, the camera pose is represented by rotation matrix R and a 3D translation vector T, which transform a point X_{wld} in world space to the camera space coordinate X_c via $X_c = RX_{wld} + T$.

2.5.2 Cost Function

Based on the extracted feature and plane tracks, we adjust simultaneously the camera poses and the parameters of the world planes and world points to make sure that the desired world planes and world points are as close as possible to the planes and points detected in each frame. The cost function to be minimized is:

$$\frac{c}{N_{pln}} \sum_{\{i,j|i\in\mathcal{D}_j\}} c_j^i D_{pln}^2 \left(Q(R_i, T_i, \mathcal{Q}_j), \mathcal{P}_j^i\right) + \frac{1-c}{N_{pt}} \sum_{\{i,k|i\in\mathcal{C}_k\}} D_{pt}^2 \left(Q(R_i, T_i, \mathbf{p}_k), \mathbf{f}_k^i\right), \tag{2.4}$$

where $D_{pln}(\cdot)$ measures the distance between a detected plane and the world plane, while $D_{pt}(\cdot)$ measures the distance between an observed feature and a world point. $Q(\cdot)$ transforms a point or a plane in the world space to a certain camera space given the camera pose. Constant c weights the effects of plane tracks against feature tracks; N_{pln} and N_{pt} are the number of planes in all plane tracks and the number of points in all feature tracks respectively. c_j^i is the weight on the plane in the plane track and equals to the number of associated pixels in a plane divided by the average number of pixels among planes in all plane tracks.

Again we use the Euclidean distance to measure D_{pt}, i.e.,

$$D_{pt}^2 \left(Q(R_i, T_i, \mathbf{p}_k), \mathbf{f}_k^i\right) = \|R_i \mathbf{p}_k + T_i - \mathbf{f}_k^i\|_2^2.$$

For distance between a detected plane and a world plane, as in Eq. 2.3, we measure the convex-hull-to-plane distance, i.e.,

$$D_{pln}^2 \left(Q(R_i, T_i, \mathcal{Q}_j), \mathcal{P}_j^i\right) = \sum_h w_{j,h}^i \|\mathbf{n}_j^T R_i^T \mathbf{v}_{j,h}^i - \mathbf{n}_j^T R_i^T T_i - d_j\|_2^2,$$

where $\{\mathbf{v}_{j,h}^i\}$ are the vertices on the convex hull of the plane $\mathcal{P}_j^i$, and $\langle \mathbf{n}_j, d_j \rangle$ are the plane

parameters of $\mathcal{Q}_j$. Putting everything together, we have the cost function,

$$\begin{aligned}&\frac{c}{N_{pln}}\sum_{\{i,j,h|i\in\mathcal{D}_j\}} c_j^i w_{j,h}^i \|\mathbf{n}_j^T R_i^T \mathbf{v}_{j,h}^i - \mathbf{n}_j^T R_i^T T_i - d_j\|_2^2 + \\ &\frac{1-c}{N_{pt}}\sum_{\{i,k|i\in\mathcal{C}_k\}} \|R_i\mathbf{p}_k + T_i - \mathbf{f}_k^i\|_2^2.\end{aligned} \tag{2.5}$$

Note that the camera pose for the very first camera is fixed at the origin with an identity rotation matrix in the above function.

Statistically speaking, the noise on observed points comes from variant independent factors, for example, errors from the 2D SIFT detector, depth map, camera calibration, thus we assume it is normally distributed based on Central Limit Theory. Hence, L2 norm is used to measure the error of an observed feature in the above cost function. As to the measurement of the error on a plane segment, we use the convex-hull-to-plane distance as a compromise since it is computationally prohibitive to use the sum of the point-to-plane distances given the number of points. As stated earlier, this convex-hull-to-plane distance is statistically the upper-bound of the error of the points on the plane segment. Since this error (or noise) also comes from various independent factors, we assume it is normally distributed and L2 norm is used.

Initialization and optimization

A general sparse Levenberg-Marquardt solver (Lourakis, 2010) is used to minimize the cost function defined in Eq. 2.5. Since there are only three degrees of freedom in a rotation matrix and two degrees of freedom in a normal vector, the Rodriguez representation and spherical coordinates are used respectively when performing the optimization. Although there are three

dataset	#frames	#features per frame	#matched features	#feature tracks	#plane tracks	#planes in tracks
SN353	228/191	328	143	6066	83/52	1179/916
SN277	381/350	497	129	10695	155/72	1896/1520
FB220	360/280	137	46	3357	138/64	1516/1232
Lab	180/180	694	267	10636	85/46	1236/858

Table 2.1: Statistics on four datasets. Under column "#frames", the number of frames in the sequence is recorded before the slash, while the number of frames registered with others is shown after the slash. The third and fourth column give the average number of all detected features and matched features respectively. Column "#plane tracks" gives the number of the detected plane tracks provided to BA and also the number of planes tracks after refinement in BA. Column "#planes in tracks" gives the total number of planes in all tracks before and after BA.

sets of unknowns, only the camera poses need to be initialized. The initial parameters for the world planes and world points are estimated from their corresponding tracks with the given camera poses. The camera poses are initialized one by one with the pairwise matching results.

Plane track refinement

Since we do not match every possible frame pair, some large planes tend to have several disjoint plane tracks instead of one complete track. These plane tracks need to be merged together. We compute the plane distance between estimated planes in the world space returned by bundle adjustment according to Eq. 2.3; if some of them are closely located, we merge them into one plane and also merge their plane tracks accordingly. Moreover, we delete a detected plane from its plane track if its distance to the corresponding world plane is beyond a threshold d. If more than half the planes of one plane track do not fit its estimated world plane, we delete the entire plane track. After refining the plane tracks, we rerun the bundle adjustment algorithm, and repeat the above procedure several times until the plane tracks no longer change.

2.6 Experiments

We evaluate our algorithm on four datasets of indoor office settings captured in real-world environment: SN353, SN277, FB220 and LAB. All the datasets except "LAB" have considerably fewer visual features. Some statistics on the datasets when running our algorithm are shown on Table 2.1. In each data set, 200 to 500 frames of RGB-D data are captured with significant overlaps. Among four datasets, "FB220" is the most challenging one, since the number of detected feature is dramatically fewer than others. On average, there are only around 10 matched points per matched frame pair (the number "46" shown under the forth column for "FB220" is the average number of the features in one frame that have matches in all other frames), which can hardly lead to robust registration (camera pose transformation between two frames) with traditional SfM methods. As shown in the second column of Table 2.1, some frames are not registered with others and thus abandoned. This is because: (1) a frame is only matched with key frames and some adjacent frames; (2) the white balance and gain of the RGB channel on Kinect is in "auto" mode and cannot be disabled with current drivers, so the appearance of objects (mainly walls and ceilings) across frames changes dramatically, especially when pointing the camera towards a light (which explains why some ceiling parts in the following results disappear), leading to missing matches between frames.

The plane/feature trade-off coefficient c in Eq. 2.5 is set to 0.1 empirically for all the experiments presented here. The registered point clouds for different rooms using our algorithm are shown in Fig. 2.1, along with the planes delivered by BA. The point clouds shown here come directly from the depth camera and are not further processed to reduce the noise, while

dataset	**point proj error**(cm)	**plane proj error**(cm)	**zero angles**(°)	**right angles**(°)
SN353	1.46± 1.37	1.74± 2.03	-0.60± 2.95	89.97± 2.14
SN277	2.01± 1.83	1.68± 1.58	-0.22± 0.80	89.94± 1.70
FB220	2.38± 1.63	2.36± 2.15	-1.17± 2.41	89.96± 2.42
Lab	1.91± 1.77	1.91± 1.66	-0.59± 4.27	89.61± 2.11

Table 2.2: The quantitative measurements of our algorithm. Inside each cell, the average error and the standard deviation are provided.

a volumetric method such as (Izadi et al., 2011; Newcombe et al., 2011) could be used to fuse all the depth map together given the camera poses.

2.6.1 Comparison to Structure from Motion (SfM) algorithms

To compare our method with state-of-the-art SfM algorithms, we ran Bundler (Snavely et al., 2007)(Snavely et al., 2006) on all datasets. Since camera poses from SfM are determined up to a scale, we need to the find this scale compared to the one used by the depth camera. SfM outputs a sparse point cloud which can be projected to image space to extract the depth values from the depth cameras. Hence by comparing the depth values from the depth camera and those from Bundler, we have the relative scale.

Not surprisingly, Bundler fails on "FB220", giving camera poses for only three frames out of 360 frames. Bundler does give results for "SN353" and "SN277", but there are dramatic misalignments between frames as shown in Fig. 2.7(a), while Bundler gives visually almost perfect results for dataset "Lab" with rich features. Clearly Bundler does not work well on dataset with relatively few features, and fails when features are very sparse. Another observation is that the frames in "Lab" have significant depth variance than frames in other three scenes. For example, in dataset "FB220" many frames only capture the side of wall, which is the degenerated case for Bundler and thus results in inaccurate camera poses.

2.6.2 Comparison to ICP method

We compared our system with RGBD-ICP algorithm (Henry et al., 2010). Since RGBD-ICP also uses visual features to constrain the planes from drifting along the plane surface direction arbitrarily, it achieved better result than Bundler. But our algorithm comfortably outperformed RGBD-ICP algorithm as shown in Fig. 2.7(b), since the error accumulation problem is not handled in RGBD-ICP, while we address that with our new bundle adjustment formulation. We also compared our system with KinectFusion(Izadi et al., 2011; Newcombe et al., 2011) which uses only the depth information to align one frame with previously accumulated data. And as expected, KinectFusion did not result in a reasonable output on our dataset either, due to its lack of constraints to handle the drifting in some frames with simple geometry structure.Since our reconstruction input frames are temporally down-sampled from the original 30 fps hand-held Kinect streams, the adjacent frames normally have noticeable amount of camera pose differences. Hence, for RGBD-ICP and KinectFusion to work, a decent initial camera pose is required. The transformation returned by our robust pairwise matching algorithm introduced in Sec. 2.4 is used for camera pose initialization.

2.6.3 Comparison to ICP with global error mitigation

As shown in (Henry et al., 2010), the accumulated error can be re-distributed globally. To fairly compare our algorithm to the ICP algorithm with error distribution, we use the same frame matching strategy, that is, to match one frame against all previous key frames and its adjacent frames. After collecting all the pairwise transformations $\{\langle R_{j\leftarrow i}, T_{j\leftarrow i}\rangle | (i,j) \in \mathcal{E}\}$,

as in (Grisetti et al., 2007), we distribute the error over the graph by minimizing

$$\sum_{(i,j)\in\mathcal{E}} \|R_{j\leftarrow i} - R_j R_i^T\|_2^2 + \|T_{j\leftarrow i} - T_j + R_{j\leftarrow i} T_i\|_2^2. \tag{2.6}$$

As shown in Fig. 2.7(c), after distributing the accumulated error, ICP gives decent results. However, some details are not preserved as well as using our method. For example, the ceiling in "SN277" is distorted as shown in Fig. 2.7(c). Note that both ICP with and without error re-distribution have to use our robust pairwise matching algorithm with planes and features for initial registration, otherwise both ICP algorithms would fail frequently in texture-less regions.

2.6.4 Quantitative measurement of errors

To evaluate our method quantitatively, we measure the projection error of feature points and planes in all datasets. The estimated 3D world points and world planes are projected to the camera coordinate system at each frame to compare them with extracted planes and features. Additionally, the relative angles between some planes in the room, such as walls, ceilings and floors, can be safely assumed to be either 0°or 90°(at least very close if not equal), and we measured these angles between the world planes delivered by our system. All the measurements are listed in Table 2.2. Although the error on the dataset "FB220" is slightly bigger than others, the projection error is considerably small and the measured angles are fairly close to 0°or 90°.

2.6.5 Running times

Our single thread program takes 2.5 to 3 seconds to extract planes from one frame, another 1.5 seconds to extract SIFT features on a desktop PC with 3.0 CPU Hz. Depending on how many planes and features are in the frames, it takes up to a few seconds to finish one pairwise matching. In our four datasets, the pairwise matching on a whole sequence takes two to five hours. We do not perform incremental bundle adjustment, and instead we perform BA on all frames directly. Generally it takes less than fifty iterations to converge with the default parameters of the chosen Levenberg-Marquardt solver (Lourakis, 2010). We run bundle adjustment and the plane track refinement repeatedly. The whole BA procedure takes 5 to 20 minutes on a dataset. We expect significant computation speed acceleration with optimized code or on parallel processing unit such as GPU. This remains one of the future directions of this research work.

2.7 Discussion

In this chapter, we present a complete pipeline of indoor 3D reconstruction with a hand-held RGB-D camera. Given indoor settings, we explore the commonly available high level plane constraint to achieve better reconstruction quality. By combining both low level feature correspondences and high level plane primitive, we significantly improve the reconstruction result in challenging cases such as surfaces lack of texture and geometry features. More specifically, we demonstrate the use of the plane primitive in robust pairwise matching even when few salient feature points are detected or they are not well distributed. This compact representation of dense points for planar parts of the scene also helps us incorporate these constraints into the

BA framework to globally mitigate the error. Real world data sets show that our method significantly improves the reconstruction quality over state-of-the-art scene reconstruction methods and the measured error is very small comparing to the ground truth.

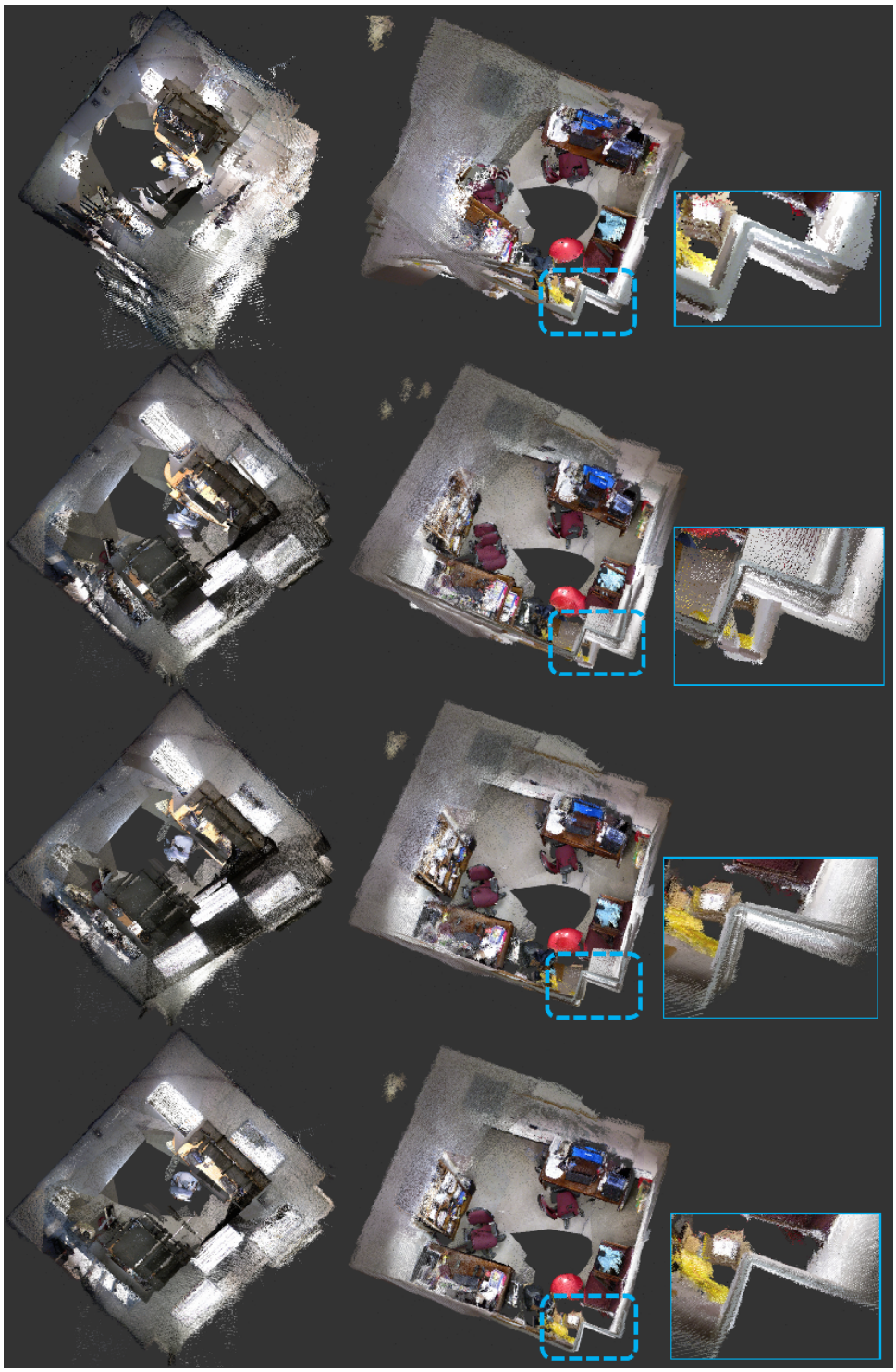


Figure 2.7: Comparison of our algorithm with other methods. The top-down view of reconstructions on two rooms are shown. 1st Row: Bundler; 2nd Row: ICP; 3rd Row: ICP+Loop Closure; 4th Row: ours. Recontruction details for one corner of the room are shown on the right.

CHAPTER 3: Nonrigid Alignment

In the following chapters, we focus on the 3D reconstruction problem for dynamic objects. As introduced in Chapter 1, we take advantage of the dynamic properties of the object being captured by accumulating data from multiple frames, which requires nonrigidly aligning data across frames. In this chapter, we will introduce the nonrigid alignment algorithm which is the fundamental component for techniques proposed in following chapters.

3.1 Nonrigid Alignment in Literature

Nonrigid Alignment is to deform a **template surface** to match with a **target surface**. It is essentially a data association/point correspondence problem. One category of methods (Nießner et al., 2013; Liao et al., 2009; Bronstein et al., 2006) directly estimates the new vertex locations of the template surface that aligns with the target surface. The problem is typically under-constrained due to the huge amount of parameters (*i.e.*, new vertex locations) and incomplete correspondence (*i.e.*, partial-to-partial matching problem), so regularization is applied to limit the deformation freedom. Nießner et al. (2013) use the As-Rigid-As-Possible constraint (Sorkine and Alexa, 2007) to enforce that each vertex moves rigidly with its neighboring vertices. Liao et al. (2009) preserve the laplacian coordinate (Sorkine et al., 2004) at each vertex during the deformation. Bronstein et al. (2006); Pokrass et al. (2013) use the isometric invariance constraint to penalize the change of geodesic distance between any points on the surface, preventing the surface from stretching and shrinking.

Another category of methods takes the indirect approach and estimates deformation parameters instead. Some work (Gall et al., 2009; Vlasic et al., 2008; Ballan and Cortelazzo, 2008; De Aguiar et al., 2008; Wu et al., 2013; Schmidt et al., 2014) assumes the articulated surface structure and use the kinematic model (*i.e.*, skeleton model) to parameterize the deformation. They typically need a manual skinning procedure, i.e., attach the pre-scanned surface (volume in case of (Schmidt et al., 2014)) to a skeleton structure or kinematic tree. The skeleton parameters are then estimated by matching the skeleton-driven surface with the current observation, such as silhouettes. A later surface refinement stage is usually employed to reduce the artifacts on the deformed surface resulting from the coarse deformation model.

The skeleton model above only applies to articulated objects, such as human beings. Sumner et al. (2007) proposed a more general motion model for shape manipulation—the embedded deformation model, which is capable of preserving geometry details while still providing desired properties such as simplicity and efficiency. Thus, it has been used to parameterize the non-rigid deformation and applied to shape matching (Li et al., 2009), which in turn is used for dynamic shape completion (Li et al., 2012; Zeng et al., 2013) and surface tracking (Bojsen-Hansen et al., 2011). Inspired by these works, we apply the Deformation Graph Model on dynamic model scanning and tracking using the noisy data from commodity depth cameras, and we incorporate both dense depth and color information into the non-rigid matching framework.

3.2 Nonrigid Alignment Problem

When deforming a template surface to match with a target surface, we want the corresponding points to be as close as possible. However, the point correspondence is unknown before the alignment. One solution is an ICP-like method (Li et al., 2009) which iteratively estimates the point correspondence and alignment parameters), but such an oscillated-optimization strategy has slow convergence and performs poorly for challenging cases. Another option is to first precompute the distance transform (*i.e.*, signed distance fields (Curless and Levoy, 1996)) for the target surface, through which the surface alignment error is measured directly without knowledge of point correspondence. Then the best alignment is found by minimizing the error with a gradient descent method. LM-ICP (Fitzgibbon, 2003) uses the same idea for rigid object alignment, and it estimates the descent direction with finite differences, which might be problematic for noisy surfaces and lead to local minima. We propose to represent the template surface as a Directional Distance Function by adding a direction field pointing to the nearest points on the surface along with the signed distance field. In this way, we can deduce an analytic solution for the derivatives of the measurement function, from which we compute the gradient descent direction. In addition, we integrate a color consistency constraint into the framework such that its derivative also has an analytical solution, which allows the problem to be solved efficiently and robustly.

3.3 Embedded Deformation Model

In general, we will want to allow our meshes to deform, for example to allow our surface reconstruction to explain the data in a depth sequence. Our desire to keep our algorithm agnostic to object class led us to choose the embedded deformation (ED) model of Sumner et al. (2007) to parameterize the non-rigid deformations of a mesh $\mathcal{V}$. In this model, a set of K "ED nodes" are uniformly sampled throughout the mesh at a set of fixed locations $\{\mathbf{g}_k\}_{k=1}^K \subseteq \mathbb{R}^3$ and neighboring ED nodes are connected to form a graph. Each vertex m is "skinned" to the ED nodes by a set of fixed weights $\{w_{mk}\}_{k=1}^K \subseteq [0,1]$, where

$$w_{mk} = (\max(0, 1 - d(\mathbf{v}_m, \mathbf{g}_k)/d_{\text{max}}))^2 / w_{\text{sum}}$$

with $d(\mathbf{v}_m, \mathbf{g}_k)$ the geodesic distance between the two, d_{max} the distance of $\mathbf{v}_m$ to its c+1-th nearest ED node, and $\frac{1}{w_{\text{sum}}}$ the normalization weight. Note $\mathbf{v}_m$ is only influenced by its c nearest nodes ($c = 4$ in our experiments) since other nodes have weights 0. The weighted deformation of the vertices surrounding $\mathbf{g}_k$ is parameterized by a local affine transformation $A_k \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t}_k \in \mathbb{R}^3$.

In addition, we follow Zeng et al. (2013); Li et al. (2013) in augmenting the deformation using a global rotation $R \in SO(3)$ and translation $T \in \mathbb{R}^3$. The precise location of vertex $\mathbf{v}_m$ deformed using the parameter set $G = \{R, T\} \cup \{A_k, \mathbf{t}_k\}_{k=1}^K$ is

$$ED(\mathbf{v}_m; G) = R \sum_{k=1}^{K} w_{mk} \left[A_k(\mathbf{v}_m - \mathbf{g}_k) + \mathbf{g}_k + \mathbf{t}_k\right] + T \tag{3.1}$$

and its associated surface normal is (with $\llbracket x \rrbracket := x/\|x\|$):

$$ED^{\perp}(\mathbf{n}_m; G) = R\left\llbracket \sum_{k=1}^{K} w_{mk} A_k^{-T} \mathbf{n}_m \right\rrbracket. \tag{3.2}$$

In addition, we allow the former functional to be applied to an entire mesh at a time to produce a deformed mesh

$$ED(\mathcal{V}; G) := \{ED(\mathbf{v}_m; G)\}_{m=1}^{M}.$$

In general, we will want to find parameters that either exactly or approximately satisfy some constraints (*e.g.* $ED(\mathbf{v}_m; G) \approx \mathbf{p}_k \in \mathbb{R}^3$), and thus encode these constraints softly in an energy function $E_{\text{con}}(G)$, *e.g.*,

$$E_{\text{con}}(G) = \|\mathbf{p}_k - ED(\mathbf{v}_m; G)\|^2.$$

In order to prevent this model from using its tremendous amount of flexibility to deform in unreasonable ways, we follow the standard practice of regularizing the deformation by augmenting $E(G)$ with

$$E_{\text{rot}}(G) = \sum_{k=1}^{K} \|A_k^T A_k - \mathbf{I}\|_F + \sum_{k=1}^{K} (\det(A_k) - 1)^2, \tag{3.3}$$

that encourages local affine transformations to be rigid, and

$$E_{\text{smooth}}(G) = \sum_{k=1}^{K} \sum_{j \sim k} \|A_j(\mathbf{g}_k - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j - (\mathbf{g}_k + \mathbf{t}_k)\|^2, \tag{3.4}$$

that encourages neighboring affine transformations to be similar. For clarity, in later equations, we use

$$E_{\text{reg}}(G) = \alpha E_{\text{rot}}(G) + E_{\text{smooth}}(G), \tag{3.5}$$

where $\alpha = 10$ in our experiments. In addition, rigidity is encouraged by penalizing the deformations at ED nodes,

$$E_{\text{rigid}}(G) = \sum_k \rho\left(\|A_k - \mathbf{I}\|_F\right) + \sum_k \rho\left(\|\mathbf{t}_k\|^2\right), \tag{3.6}$$

where $\rho(\cdot)$ is a robustness kernel function.

To summarize, nonrigid alignment paramters are estimated for the deformation graph by solving,

$$\min_G w_{\text{reg}} E_{\text{reg}}(G) + w_{\text{rigid}} E_{\text{rigid}}(G) + w_{\text{con}} E_{\text{con}}(G) \tag{3.7}$$

It is straightforward to minimize this energy using standard nonlinear least squares optimization (Sumner et al., 2007; Li et al., 2009). The third term E_{con} comes from the matched key points of two surfaces, In our case, the key points are Lucas-Kanade corner points that are converted to 3D points from 2D image locations using their observed depth.

The energy terms above are inadequate to align noisy 3D data from commodity depth cameras. Therefore, we include two more terms—dense point cloud alignment $E_{\text{dns\_pts}}$ (Section 3.4)

and color consistency E_{clr} (Section 3.5), transforming Equation 3.7 to

$$\begin{aligned}\min_{G}\ & w_{\text{reg}}E_{\text{reg}}(G) + w_{\text{rigid}}E_{\text{rigid}}(G) + w_{\text{con}}E_{\text{con}}(G) \\ & + w_{\text{dns\_pts}}E_{\text{dns\_pts}}(G) + w_{\text{clr}}E_{\text{clr}}(G)\end{aligned} \tag{3.8}$$

3.4 Directional Distance Function and Measurement of surface alignment

The matched key points in Sec. 3.3 are sparse features on the surface; their alignment does not represent the global surface alignment. Thus the dense alignment must be measured. Different from Li et al. (2009) who iteratively estimates dense point correspondence, we represent the target surface as a distance field so that the surface alignment can be efficiently measured. At each voxel of this volume data, we record its distance $\mathcal{D}(\cdot)$ and direction $\mathcal{P}(\cdot)$ to its closest point on the surface. This representation is an extension to the Signed Distance Function (SDF) (Curless and Levoy, 1996), and we call it the Directional Distance Function (DDF). Then, the energy function for dense point cloud alignment is defined by,

$$E_{\text{dns\_pts}}(G) = \sum_{i} \big(\mathcal{D}(\tilde{v}_i)\big)^2 \tag{3.9}$$

where $\tilde{v}_i = ED(v_i, G)$.

Calculating the DDF takes no more effort than recording the position of the nearest point on surface and subtracting its position to get $\mathcal{P}(\cdot)$. The pseudo-code of calculating the DDF from a surface or depth map is given in Appendix A. Note that the voxel whose closest surface point lies at the boundary of an open surface is set to null (or empty), which prevents the undesired surface extension when recovering a triangle mesh from a DDF. The surface boundary

is identified either as pixels on a depth map that have depth discontinuity with their neighbors or the vertices on a triangular mesh that do not share its edge with other triangles.

$\mathcal{P}(\cdot)$ in the Directional Distance Function is especially helpful when minimizing Eq. 6.3. Since the energy function is in least squares form, it can be efficiently solved via a gradient descent-like method (e.g. Gauss-Newton algorithm) as long as the Jacobian matrix J is provided. To solve this nonlinear least squares problem, we use the Levenberg-Marquardt algorithm (Madsen et al., 2004) implemented by the Google Ceres solver (Agarwal and Mierle, 2013). The Jacobians for first three terms of Eq. 6.3 are straightforward; we will illustrate those for E_{dns_pts} here and E_{clr} in next section. One interesting fact about SDF $\mathcal{D}(\cdot)$ is that its gradient is a unit vector aligned with the direction pointing to the closest surface point, i.e., $\mathcal{P}(\cdot)$. More precisely, since we use a negative SDF for voxels inside a surface,

$$\nabla\mathcal{D} = \begin{cases} -\mathcal{P}, & \text{if } \mathcal{D} > 0 \\ \mathcal{P}, & \text{if } \mathcal{D} < 0 \\ \text{normal}, & \text{if } \mathcal{D} = 0. \end{cases} \tag{3.10}$$

Thus, the Jacobians $\frac{\partial}{\partial p_k}\mathcal{D}(\tilde{v}_i) = \nabla\mathcal{D}|_{\tilde{v}_i}\frac{\partial}{\partial p_k}\tilde{v}_i$, where p_k is the k-th deformation parameter. When computing DDF in practice, we align $\mathcal{P}$ to surface normal when $|\mathcal{D}| < \epsilon$, which is analogous to using the point-to-plane distance instead of the point-to-point distance for ICP. In our implementation, ϵ is set to 1.5 cm.

Sometimes there are parts on the target surface where front and back surfaces are close enough that the deformed surface point $\tilde{v}_i$ is attracted to the wrong surface during iterations.

Fortunately, $\nabla\mathcal{D}|_{\tilde{v}_i}$ is the approximation of the normal of $\tilde{v}_i$'s closest point on the target. When the normal $\tilde{n}_i$ on the deformed reference surface does not agree with $\nabla\mathcal{D}|_{\tilde{v}_i}$, this means $\tilde{v}_i$ is heading to the wrong side of the target surface. To resolve this, we let $\frac{\partial}{\partial p_k}\mathcal{D}(\tilde{v}_i) \leftarrow 0$, if $\nabla\mathcal{D}|_{\tilde{v}_i} \cdot \tilde{n}_i < 0$, nullifying the attraction from the wrong part of the target.

3.5 Color Consistency

When deforming the template surface to the target, the matched points must have similar color (or texture). The E_{clr} term helps resolve alignment ambiguities when a near-symmetric part on the surface rotates or moves, such as head turns and arm rotations. In our scanning and tracking system, the template surface is the currently accumulated 3D model from the depth and color of previous frames, and it is represented by a triangle mesh with a 3D color vector c_i attached at each vertex. The target surface is the current observation of the dynamic object, and its raw representation is a set of depth maps $\{Z_k(\cdot)\}$ and color images $\{I_k(\cdot)\}$.

All the depth and color cameras are calibrated under the same world coordinate system, and $P_k(\cdot)$ projects a 3D point to the k-th image coordinate. Thus, the color consistency term in Eq. 6.3 is

$$E_{\text{clr}}(G) = \sum_k \sum_i \left\| \delta_k\left(\tilde{v}_i\right) \cdot \left[I_k\left(P_k\left(\tilde{v}_i\right)\right) - c_i \right] \right\|_2^2, \tag{3.11}$$

where $\delta_k\left(\tilde{v}_i\right)$ is the visibility term; $\delta_k\left(\tilde{v}_i\right) = 1$ when $\tilde{v}_i$ is visible to the k-th color camera, and 0 when invisible. Visibility checking is performed with a z-buffer algorithm—project the surface to the image space and the vertices with smallest z values are visible. We also set $\delta_k = 0$ for vertices whose outward normals point away from the camera, to prevent holes in

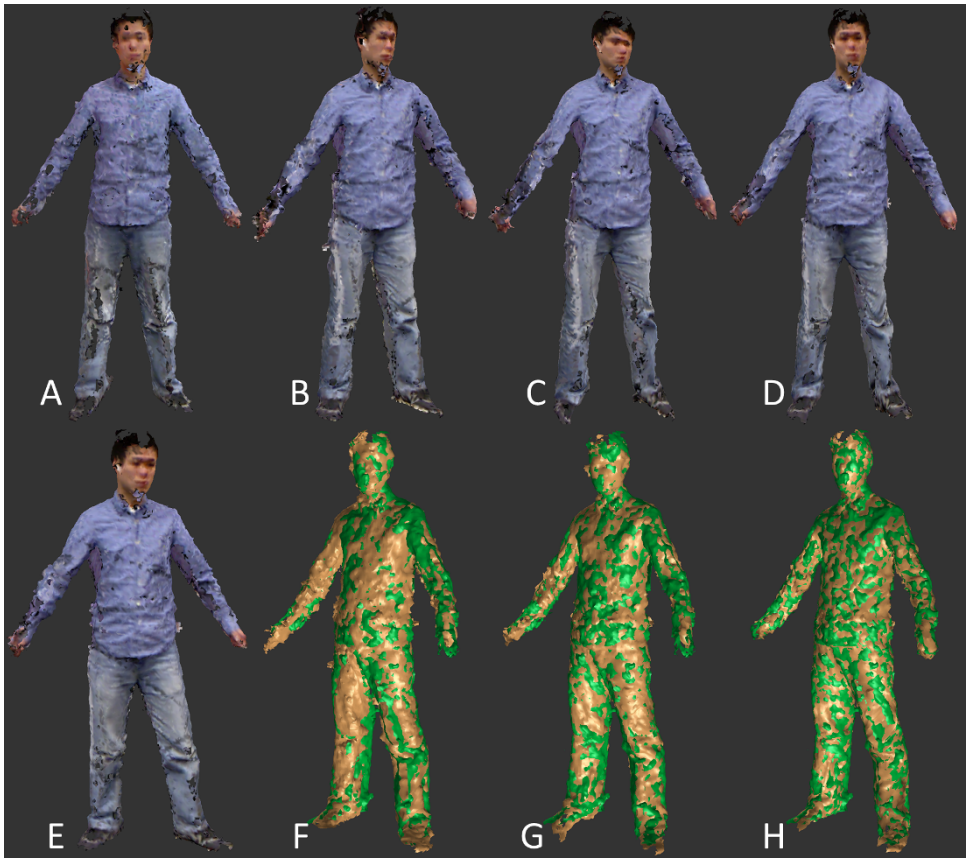
the incomplete front surface from erroneously letting parts of back-facing surfaces pass the z-buffer test.

The Jacobians for E_{clr} also have an analytic solution: $\frac{\partial}{\partial p_i} I_k^c \left(P_k \left(\tilde{v}_i \right) \right) = \nabla I_k^c \cdot \frac{\partial}{\partial \tilde{v}_i} P_k \cdot \frac{\partial}{\partial p_i} \tilde{v}_i$, where ∇I_k^c is the image gradient for the c-th channel of the k-th color image. Note that the visibility check needs to be performed at each iteration of the gradient descent method since each iteration produces a differently deformed template surface. Fortunately, it does not take much more effort since we need to project vertices to image spaces anyway.

This color consistency term essentially estimates the optical flow that matches 3D surface points to 2D image coordinates, while classical 2D optical flow performs matching in the image space (Baker and Matthews, 2004). Even though the 3D flow technique (Herbst et al., 2013) estimates the dense 3D motion with RGB-D data, the matching is still confined in the 2D image space. In addition, our algorithm does not require that the color and depth image be aligned, making it possible to use high resolution color cameras.

3.6 Comparison

Figure 3.1 compares our algorithm with previous works from Sumner et al. (2007) (designed for shape editing and used by others for non-rigid alignment) and Li et al. (2009) and shows the results of aligning one frame to another. Sumner et al. (2007) only uses matched sparse points for alignment (E_{con} in Eq. 6.3), so the dense points are not perfectly aligned. Li's algorithm performs better since dense points is employed for alignment; as shown in Figure 3.1(G), the reference and target surfaces after alignment are well interleaved compared with (F). But the textures of aligned surfaces do not match for both algorithms; in Figure 3.1, neither



Source/Target Sumner *et al.* Li *et al.* Ours

Figure 3.1: Comparison of various algorithms aligning the source surface (E) to the target surface (A). (B)(C)(D) show the deformed source surfaces with three algorithms (Sumner *et al.*, Li *et al.*, and ours) respectively; (F)(G)(H) show deformed source surfaces over target surfaces where the source is in green and the target in orange. Note neither method of Sumner *et al.* nor that of Li *et al.* is able to align the head part correctly, while our method gives exact alignment by using color consistency constraints.

(B) nor (C) is close to (A). Since our algorithm includes a color consistency constraint, it performs better than the other algorithms. Not only do the surfaces align more tightly, the textures match as well. Quantitative and qualitative evaluation on our nonrigid alignment algorithm are shown in later chapters in more specific contexts.

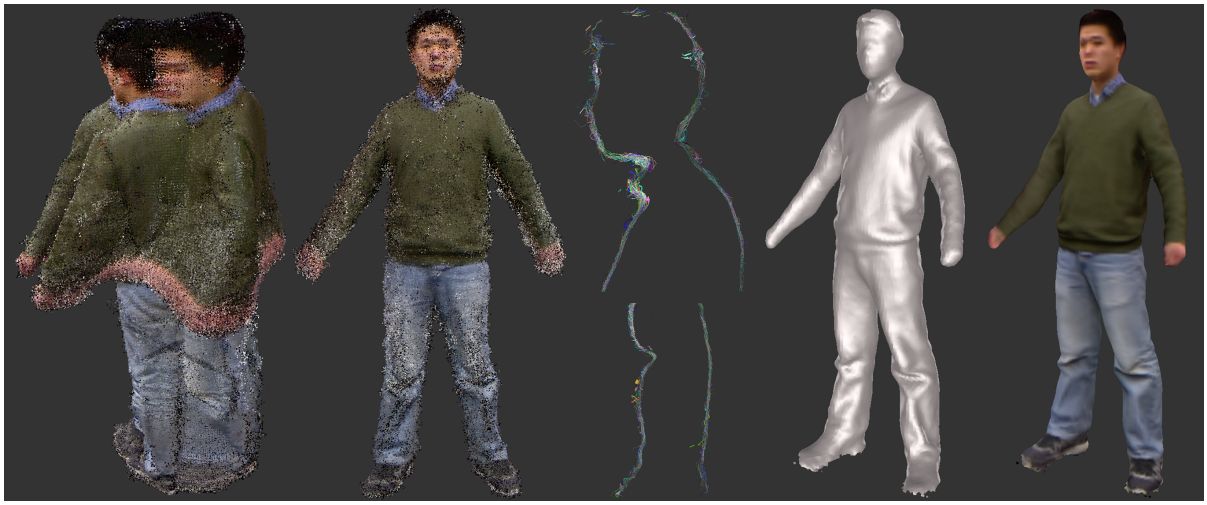
CHAPTER 4: Dynamic Surface Reconstruction

In this chapter, we introduce a 3D capture system that first builds a complete and accurate 3D model for dynamic objects (*e.g.*, human body) by fusing a data sequence captured by multiple commodity depth and color cameras (*e.g.*, Microsoft Kinects), and then tracks the fused model to align it with following captures. One crucial component of our system is the nonrigid alignment which has been introduced in Chapter 3. Our system also extends the volumetric fusing algorithm (Curless and Levoy, 1996; Newcombe et al., 2011) to accommodate noisy 3D data during the scanning stage. We introduce the scanning system in Sec. 4.2, and the tracking system in Sec. 4.3.

4.1 Related work

KinectFusion shows successful scanning of static scenes in real-time in Newcombe et al. (2011) and shows results on processing dynamic objects with piece-wise rigid matching in Izadi et al. (2011). Our work aims at scanning and tracking dynamic objects by employing a nonrigid alignment algorithm.

Previous work on fusing multiple scans of a dynamic object either restricts the motion of the object (*e.g.*, by using a turntable to rotate an object in a predefined manner (Tong et al., 2012)), or uses a human model database (e.g., SCAPE (Anguelov et al., 2005)) as prior shape knowledge for the human body (Hirshberg et al., 2012; Weiss et al., 2011). This gives up the flexibility to scan other objects or humans wearing loose clothes. Our scanning system allows



(a) Pre-aligned Point Clouds (b) Post-aligned Point Clouds (c) Cross Section of alignment (d) Fused Scan (e) Fused Scan with Color

Figure 4.1: Scanned Dynamic Objects. (a)&(b) show 150 frames of point clouds of one moving person before and after alignment. (c) shows the cross section of the aligned surfaces at the upper body and leg (surfaces are assigned different colors). (d)&(e) show the fused 3D model.

for large deformations and motion of the subjects being scanned and makes no assumption on their shapes.

Our work also relates to research into motion/performance capture of articulated dynamic objects (Gall et al., 2009; Vlasic et al., 2008; Ballan and Cortelazzo, 2008). These works presume a template surface model of the object being tracked is available. They also need a manual skinning procedure, i.e., attach the pre-scanned surface to a skeleton structure or kinematic tree. De Aguiar et al. (2008) abandoned the skeleton and uses a coarse tetrahedral version of the scan instead, but it still involves manual procedures during initialization. Our system instead incorporates a scanning procedure that takes advantage of depth sensors and does not need the manual operations such as skinning or rigging. While the systems above typically need to use a blue-screen to get the object silhouette, our system has no such requirement.

Our work relates to the research into 3D Character Animation (Baran and Popović, 2007; Chen et al., 2012) as well. Baran and Popović (2007) automatically fit a skeleton to the char-

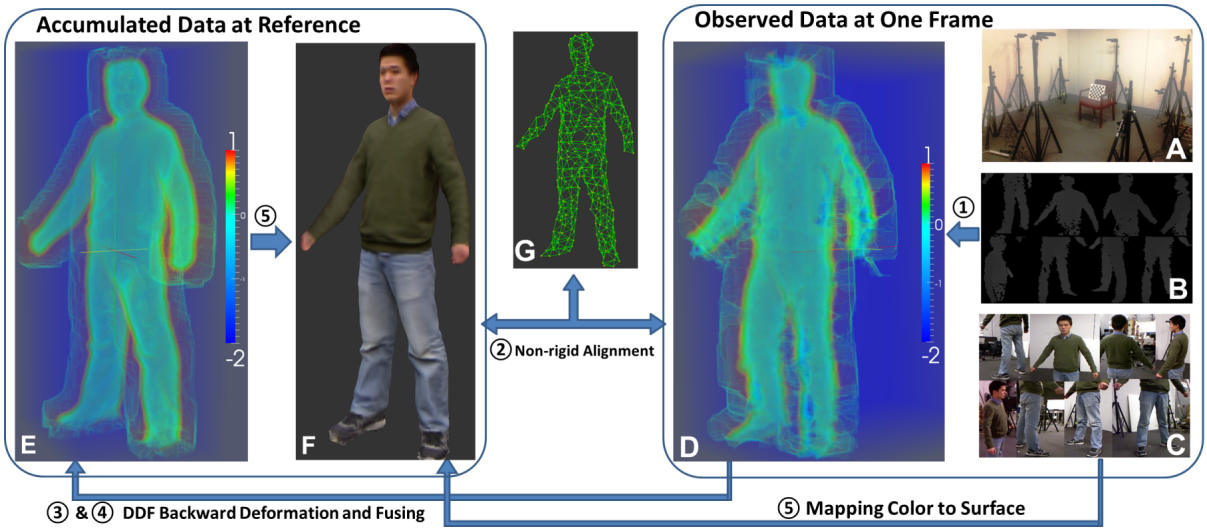


Figure 4.2: The Scanning System. ①–⑤ the steps in Algorithm 1; (A) the Eight-Kinect System Setup; (B) depth maps; (C) color images; (D) volume visualization of the fused DDF from 8 depth maps; (E) volume visualization of the accumulated DDF at the reference so far; (F) the accumulated reference surface so far; (G) the deformation graph on the surface (F).

acter mesh and attach it to the surface. Chen et al. (2012) designed a real-time system that transfers user's motion to any pre-scanned object. The user's skeleton is tracked from a Kinect camera and is attached to the embedded deformation graph (Sumner et al., 2007) of the character, and the character is then deformed accordingly. Instead of only transferring user's motion, we aim at a tight surface alignment between a user's pre-scan and the later observation. We also work without explicit requirement to track human skeletons allowing support for arbitrary objects.

4.2 Scanning Dynamic Objects

The objective for scanning a dynamic object is to acquire a complete and accurate model from a sequence of depth maps and color images captured by several commodity depth and color cameras. In our system, eight Kinects are used and placed in a circle with radius of 1.8

Algorithm 1: Scanning Pipeline

Set the data from the first frame as the reference;
foreach *new frame* **do**
- **1.** fuse depth maps to get the DDF;
- **2.** match the reference surface to the new observation;
- **3.** transform the DDF from target to reference;
- **4.** fuse the transformed DDF into MDDF at reference;
- **5.** generate the reference surface from MDDF, and map colors to the surface;

meters. Four of them cover the upper space, and the other four cover the lower space. Our system setup is shown in Fig. 4.2(a).

We use a similar data fusion pipeline as KinectFusion (Izadi et al., 2011; Newcombe et al., 2011). Initially, the data of the first frame is set as the reference; then we repeatedly estimate the deformation between the reference and newly observed data, and fuse the observed data to the reference. More specifically, we perform following steps for each newly observed frame:

1. Convert depth maps from the Kinects to Directional Distance Functions (DDFs, introduced in Chapter 3.4), and fuse them into one DDF $\mathcal{F}_{trg}$ using the method introduced later in Section 4.2.2.

2. Sample an Embedded Deformation Graph (ED, introduced in Chapter 3.3) from the reference surface, and estimate its parameters with the nonrigid alignment algorithm introduced in Chapter 3 to align the reference surface with the new observation ($\mathcal{F}_{trg}$ and color images). The parameters for this forward deformation is computed.

3. Compute the backward deformation (from target to reference) according to the forward deformation, and transform the fused DDF from step 1 to the reference, i.e., $\mathcal{F}_{trg} \rightarrow \mathcal{F}_{ref}$. Details are provided in Section 4.2.1.

4. Fuse $\mathcal{F}_{ref}$ into the Multi-Mode Directional Distance Function (MDDF) at reference. Unlike DDF, MDDF has multiple distance values and direction vectors at each voxel. A detailed introduction of MDDF and fusion of multiple DDFs is presented in Section 4.2.2.

5. Finally, generate the reference surface from the MDDF and texture it by all color images observed so far. To texture the surface, we deform the surface to various frames according to the estimated forward deformations and project each vertex to the image spaces, the color pixels the vertex falls on are averaged to obtain one 3D color vector.

The scanning procedures are summarized in Algorithm 1, and a graphical illustration is given in Figure 4.2. Note that we always align the fused data to following frames, which helps to deal with the error accumulation problem (Izadi et al., 2011; Newcombe et al., 2011). Also note that we do not directly estimate the backward deformation parameters by deforming the target to the reference. This avoids generating an embedded deformation graph from the noisy surface model at target.

4.2.1 DDF Transformation from Target to Reference

Given the forward deformation parameters $\{\langle A_j, t_j \rangle\}$, we could set the backward deformation parameters as $\{\langle A_j^{-1}, -t_j \rangle\}$ and the graph node position as $g_j + t_j$; however it does not guarantee a backward alignment, since the inverse of the linear interpolation of matrices $\{A_j\}$ does not equal the linear interpolation of the inverse of these matrices. Instead, we find the point correspondence of the reference and target according to the forward deformation, and estimate the backward deformation parameters via Eq. 3.7 by formulating the point correspon-

Algorithm 2: DDF transformation $\mathcal{F}_{trg} \rightarrow \mathcal{F}_{ref}$

foreach *i-th voxel of $\mathcal{F}_{trg}$ at location p_i with direction to nearest surface point denoted as P_i and distance value as D_i* **do**

1. find p_i's nearest point on target surface whose neighboring ED nodes are used for later deformation;

2. deform its location according to Eq. 3.1: $p_i \rightarrow \tilde{p}_i$;

3. deform its direction P_i according to Eq. 3.2: $P_i \rightarrow \tilde{P}_i$;

4. record the deformed voxel as a 4-tuple $\langle p_i, \tilde{p}_i, \tilde{P}_i, D_i \rangle$;

foreach *each voxel of $\mathcal{F}_{ref}$ at location q* **do**

1. find the set of its neighboring deformed $\mathcal{F}_{trg}$ voxel:
$\mathbb{S} = \{\langle p_k, \tilde{p}_k, \tilde{P}_k, D_k \rangle|_{\|\tilde{p}_k - q\| < \epsilon}\}$;

2. Divide $\mathbb{S}$ in to subgroup $\{\mathbb{G}_i\}$ by clustering on p;

3. Find the subgroup $\mathbb{G}_s$ with smallest averaged D;

4. set the direction and distance value of $\mathcal{F}_{ref}$ at q:
$P^{ref} = \sum_{k \in \mathbb{G}_s} w_k \tilde{P}_k$
$D^{ref} = \sum_{k \in \mathbb{G}_s} w_k D_k$
where $w_k = exp(-(q - \tilde{p}_k)^2)$

dence into E_{con}.

With the backward deformation estimated, one way of deforming the DDF is first generating the underlying triangulated surface from the DDF, then deforming the surface, and eventually re-computing the new DDF from the deformed surface. This method seems plausible, but the DDF field structure is not preserved during transformation due to downgrading the DDF to a surface. We choose to directly apply deformation on DDF. Although the nonrigid transformation is only defined on the surface, each voxel of the DDF could be transformed according to the deformation parameters of its closest point on the surface. Alg. 2 shows our solution of deforming DDF $\mathcal{F}_{trg}$ to $\mathcal{F}_{ref}$ according to the deformation graph $\mathbb{G}$. Note that we need to handle the situation when the transformed voxels collide with each other. At each voxel position on $\mathcal{F}_{ref}$, we find its nearby transformed $\mathcal{F}_{trg}$ voxels, then group them according to their original grid positions, and find the group with smaller absolute distance value, from

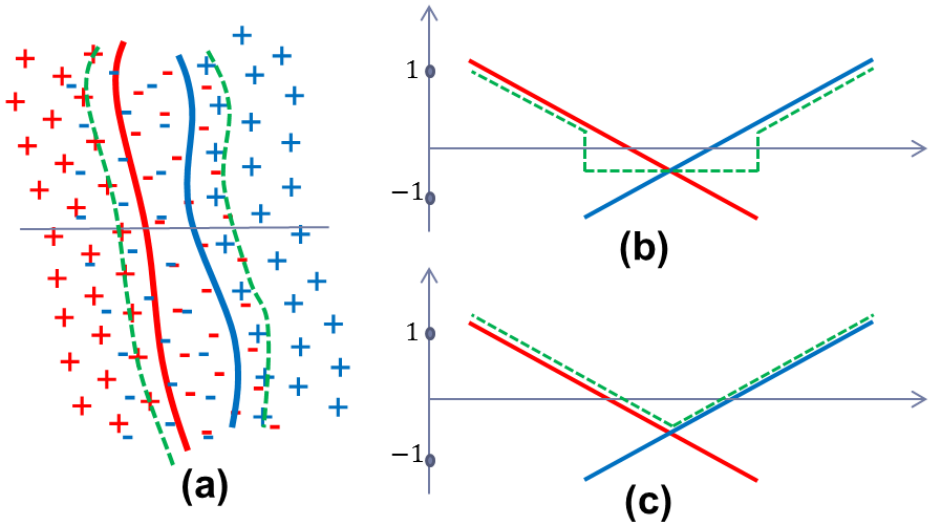


Figure 4.3: Fusing two 2D truncated signed distance fields. (a) shows two curves and their signed distance fields in red and blue respectively. Directly adding two signed distance fields ends up with expanded curves shown in green. (b) shows the signed two distance functions along one line across two curves in red and blue, and the sum of the two in green. (c) shows the above expansion effect is prevent by taking the one with smaller absolute value.

which the direction vector and signed distance value for the $\mathcal{F}_{ref}$ voxel are interpolated.

4.2.2 Fusion of multiple DDFs

When the noise level of the 3D data is low, summing over multiple aligned DDFs cancels out the noise, as shown in the work of KinectFusion (Izadi et al., 2011; Newcombe et al., 2011). Then the surface can be recovered by finding the zero-crossing of the fused distance field using algorithms such as marching cubes (Lorensen and Cline, 1987). However, when the noise level is as large as the object dimension, summing over the distance field raises problems as illustrated in Fig. 4.3(a)&(b). This is because the distance field needs to be truncated so that the distance field of a front surface does not interference with the surface behind. The distance behind the surface where truncation begins, denoted as μ, should be proportional to the noise

level. Unfortunately, in the case of Fig. 4.3(a), when a big μ is chosen to suppress the noise, the zero-crossing of the fused distance field does not align with the surface anymore due to the interference between the distance functions of the front and back surface. This is exactly the case when performing dynamic object scanning via commodity depth cameras, since the noise is evident in the depth maps, due to sensor noise, calibration error and nonrigid alignment, and it can easily go beyond the dimension of thinnest part of the object, such as a palm or wrist.

Fortunately, in many cases, $\nabla\mathcal{D}$ differentiates which surface a distance value corresponds to. And in our DDF representation, $\nabla\mathcal{D}$ can be easily obtained using Eq. 3.10. When fusing DDFs, at each voxel, we only sum over the distance value $\mathcal{D}$ with similar $\nabla\mathcal{D}$, preventing the interference between the distance fields corresponding to different surface parts. This results in a new data structure: Multi-Mode Directional Distance Function (MDDF). Each voxel of a MDDF records a set of averaged $\mathcal{D}$'s, $\nabla\mathcal{D}$'s, and the weights on all modes. In our scanning Step 4, a new $\mathcal{F}_{ref}$ can be easily fused to MDDF. First, the mode with most similar $\nabla\mathcal{D}$ is found at each voxel; then its distance value and $\nabla\mathcal{D}$ is incorporated to that mode and the weights are updated.

To recover the underlying surface from a MDDF, it needs to be converted to a DDF. One mode is selected among all the modes at a MDDF voxel as the DDF voxel. We choose the one with smaller absolute distance value. As illustrated in Fig. 4.3(c), this strategy solves the interference problem shown in Fig. 4.3(b). In practice, the mode with relatively smaller weights (50% of the largest weight in our experiment) are discarded during the above conversion.

4.3 Dynamic Surface Tracking

After fusing a number of DDFs (a few hundreds in our case), the improvement on a scanned model tends to converge. Thus, after a complete model is achieved, we deform the scanned model to the current depth and color observations, i.e., only Steps 1 and 2 in Algorithm 1 are used during this stage—tracking. Note that a fixed Deformation Graph structure is used in Step 2 during tracking.

To track fast moving surfaces, we use a Kalman Filter to predict the translation vector t_j for each deformation graph node of the next frame , and use its prediction as the initial parameter of the non-rigid alignment problem. The matrices $\{A_j\}$ are simply initialized using the values of the last frame.

4.3.1 Tracking Surfaces with Isometric deformations

In many cases, the surface being tracked is roughly under isometric deformations (Bronstein et al., 2006), i.e., the geodesic distance of any pair of surface points is preserved during deformation. For example, the deformation of the 3D human body model is near isometric, if not strictly isometric. Thus, for these cases, we add a new term E_{len} to our energy minimization problem in Eq. 6.3,

$$E_{\text{len}}(G) = \sum_{j=1}^{J} \sum_{k \in \mathbb{N}(j)} \| |g_j + t_j - g_k - t_k| - |g_j - g_k| \|_2^2 . \tag{4.1}$$

where g_j and t_j are the node location and translation vector of the Deformation Graph respectively, and $\mathbb{N}(j)$ are the neighbors of the j-th node. E_{len} penalizes the changes of the length

of the edge connecting the neighboring nodes during deformation. Although E_{len} does not guarantee an exact isometric deformation, in practice it works well to prevent the surface from stretching or shrinking. In our implementation, we use the robust estimation technique (Agarwal and Mierle, 2013) on E_{len} to allow for length changes for some parts (outliers).

4.4 Experimental Results

To test our system, we captured several sequences of people performing various movements using the eight Kinects setup shown in Fig. 4.2(A). Both depth maps and color images from Kinects are used for nonrigid matching, and both have a resolution of 640×480. Since the depth coming from Kinects deviates from the true depth value (Beck et al., 2013), we correct this depth bias using a linear mapping function for each Kinect separately.

The resolution of the lattice of the volumetric representation (DDF) is set to 1cm in our experiment to reduce the processing time and memory, yet it is still enough to output relatively high quality models. The DDF with higher resolution is necessary to achieve the quality of a high-grade commercial laser scanner. In all of our experiments, μ in DDF is set to 4cm; the weights in Eq. 6.3 are set as follows, $w_{rot} = 30$, $w_{reg} = 5.0$, $w_{con} = 1.0$, $w_{dns\_pts} = 5.0$, $w_{clr} = 5.0$, and $w_{len} = 3.0$. These parameters are chosen experimentally. Both the color intensity and signed distance value have been scaled to $[0, 1]$. The Google Ceres solver generally converges after 10 iterations on Eq. 6.3. The Deformation Graph Model is uniformly sampled on the surface based on the geodesic distance. The minimum distance between neighboring nodes is set to 7cm, and about 350 graph nodes are sampled. A denser graph model does not show visually improved results.

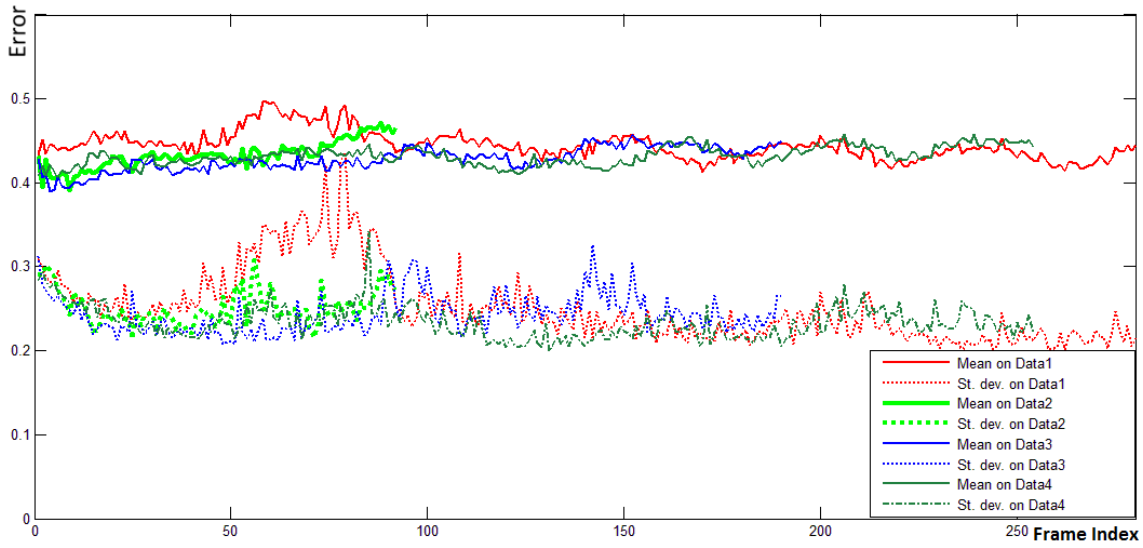


Figure 4.4: Scanning Errors (in cm). The solid lines indicate the mean deviation of the fused surface to the observed surface at each frame on four data sequence. The dashed line shows the standard deviation of the error.

During scanning or tracking, around 150 pairs (per image pair) of matched corner points from the current and previous frame are found via LK optical flow. The points of the previous frame are transformed to the reference via the backward deformation. Note that E_{dns_pts} and E_{clr} only work when the initialization is reasonably close to the optimal solution. Thus, when the object moves fast and the initial is far off the optimal, E_{con} plays an important role. Otherwise, it is overwhelmed by E_{clr}—dense color matches. In our experiment, we ignore E_{con} in our tracking state to save us from computing the backward deformation and use the Kalman Filter to predict a reasonable initial value.

4.4.1 Scanning

Some scanned 3D models of full human bodies are shown in Fig. 4.6. Despite the low DDF resolution, our algorithm still recovers the geometry details such as wrinkles on the clothes.

Figure 4.5: Intermediate Scans at frame 1, 5, 15, 30, 50, 80, and 150.

Fig. 4.6 also shows the color on the vertices of the model averaged from all the frames used for fusing (200~300 frames). The sharpness of the color indicates that the frames are well aligned. Be aware that the sparseness of the model vertices (due to the low DDF resolution) leads to some blur from the color interpolation during GPU rasterization. A few intermediate models created during scanning are shown in Fig. 4.5. The noise is gradually filtered out and holes filled up while the geometry details are preserved when accumulating more and more frames.

We evaluated the scanning system quantitatively. The deviation of the scanned model from the observed surfaces at each frame is shown in Fig. 4.4. To measure the deviation, the observed surfaces are deformed to the reference as shown in Fig. 6.1(b). The averaged distance between matched points is used as an measurement of the surface deviation. As shown in Fig. 4.4, the deviation stays around 0.42 cm for all frames of all test data sequences, indicating our nonrigid

Figure 4.6: Scanned Surfaces Generated by Our System.

alignment algorithm works decently and the scanning system handles the error accumulation problem well.

4.4.2 Tracking

We test our tracking algorithm on multiple sequences with people performing various movements. Fig. 4.7 shows some results of tracking the scanned surface. The robust nonrigid matching algorithm enables our system to track difficult gestures involving significant topology changes such as arm folding or fast surface deformation such as squatting and stretching.

4.5 Discussion

Our work shows temporal information of depth and color is helpful for modeling dynamic objects. We demonstrate a complete and accurate surface is fused from a sequence of depth and color data from commodity depth and color cameras. By tracking this fused surface over time, we acquire an improved 3D model for later frames. The key components of our system are a nonrigid alignment algorithm that integrates both depth and color information and a volumetric fusion algorithm that handles large noise on depth observations.

4.5.1 Limitations of the System

Synchronization The synchronization across Kinects is still an unsolved problem, but the current setup works reasonably well when the subject of interest moves naturally. We expect this problem to be eventually solved by hardware.

Segmentation. Currently, to segment the foreground dynamic objects from the rest of the

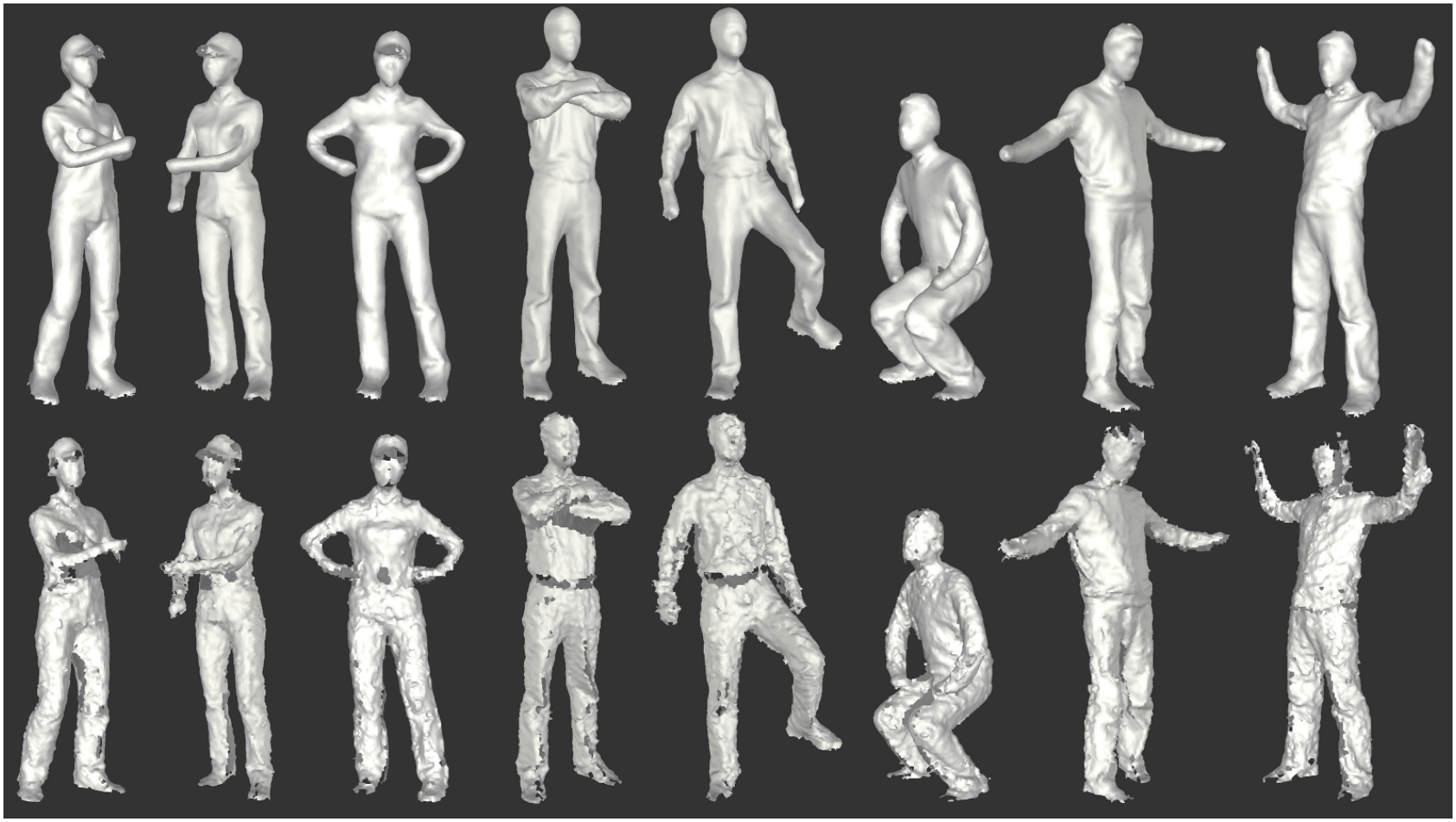


Figure 4.7: Surface Tracking. The top row shows the deformed human scan; the bottom row shows the input surfaces from Kinects.

scene on the depth map, we first eliminate the floor area via a clipping plane and other background outside a bounding box, then perform connected component labeling to filter out small noisy blobs. The depth-only segmentation works fine on the human body except the feet. The problems on the feet are visible in the scanned models in Fig. 4.6. An improved segmentation might employ color information to refine the boundary.

Topology Change. During the scanning stage, we ask the people being scanned to turn around, performing any movement they want except those changing the body topology such as crossing the arms. Our scanning algorithm assumes the body topology does not change. This requirement could be removed by identifying the body parts with topology changes in each frame and only fusing the body part without topology changes. The tracking stage remains free of these restrictions; we make no assumptions on body topology then.

Processing Time. Most of the system implementation is single-threaded and performed on CPU. As a result, it takes approximately one minute per frame to fuse or track one frame containing approximately 30,000 vertices. Most of the computation time is dedicated to solving Eq. 6.3. By taking advantage of the GPU technology and multi-grid computation strategy as KinectFusion does, we expect boosted performance in our system.

CHAPTER 5: Dynamic Surface Recontruction with a Single Depth Sensor

In the previous chapter, we achieve dynamic object scanning with multiple units of depth sensors, which requires careful calibration and thus limits its application for a non-technical user. In this chapter, we investigate dynamic object reconstruction with a single depth sensor.

KinectFusion (Izadi et al., 2011; Newcombe et al., 2011) achieves high quality scanning by moving a Kinect sensor around an object. But it uses a rigid alignment algorithm and thus requires the object or scene being scanned to remain static. In some scanning applications such as human body scanning, particularly of children, nonrigid movement is inevitable. Also in-hand scanning of soft objects introduces deformations when the objects are rotated to get a complete scan. To address these issues we present a new 3D scanning system for arbitrary scenes, based on a single sensor, which allows for large deformations during acquisition.

Our goal is to combine a sequence of depth images, each representing a noisy and incomplete scan of the object of interest, into a high quality and complete 3D model (Fig. 6.1). Even for rigid alignment, the problem of *drift* occurs when aligning a sequence of partial scans consecutively, where the alignment error accumulates quickly and the scan does not close seamlessly (Fig. 5.3(b)). KinectFusion alleviates some level of drift by aligning the current frame with the fused model instead of the previous frame (Newcombe et al., 2011). Note drift is not a concern in our earlier system given that a relatively complete model is captured at each frame.

Many follow-up systems based on KinectFusion have specifically looked at scanning humans (e.g., for 3D printing or generating avatars) where the user rotates in front of the Kinect

Figure 5.1: A mother holds an energetic baby while rotating in front of a Kinect camera. Our system registers scans with large deformations into a unified surface model.

while maintaining a roughly rigid pose, e.g., (Weiss et al., 2011; Li et al., 2013; Tong et al., 2012; Zeng et al., 2013; Helten et al., 2013). This highlights the fundamental issue when scanning living things – they ultimately move.

To make this problem more tractable, some systems make strong assumptions about the nonrigid object being a human, using either parametric models (Weiss et al., 2011; Helten et al., 2013) or limiting the user to certain poses such as a 'T' shape (Cui et al., 2013). We wish to avoid such scene assumptions. Li et al. (2013) adapt a more general nonrigid registration framework which can support a wider range of poses, clothing or even multiple users. This system demonstrates compelling results but relies on a very specific type of user interaction: the user moves in roughly 45 degree increments, in front of Kinect, and at each step remains static, whilst the motorized sensor scans up and down. Each of these fused static scans are then nonrigidly registered and a global model reconstructed. Here the user is assumed to explicitly perform a loop closure at the end of sequence. For certain subjects, such as children or animals,

such loop closure will often happen less explicitly, many times over a single sequence.

Zeng et al. (2013) show that when using nonrigid alignment to an embedded deformation (ED) model (Sumner et al., 2007) for *quasi-rigid* motion, drift is greatly alleviated, and loop-closure can be made *implicit*. However, for nonrigid motion, our experience (Fig. 5.7) shows that drift is still a serious problem even when scanning mildly deforming objects such as a turning head.

To deal with severe drift, loop closure must be explicitly dealt with, but without restricting user motions. However, dealing with such loop closures, is only one piece of the puzzle, as this only evenly distributes error over the loop instead of minimizing the alignment residual. Thus, our pipeline also performs a dense nonrigid bundle adjustment to simultaneously optimize the final shape and nonrigid parameters at each frame. We use loop closure to provide the initialization for the bundle adjustment step. Our experiments show that bundle adjustment gives improved data alignment and thus a high quality final model. We also show that the bundle adjustment is able to converge to a reasonable solution even though loop closure fails in some cases.

We will summarize previous work in the next section and describe our surface in Sec. 5.2. From Sec. 5.3 through Sec. 5.5, we explain the preprocessing procedures for bundle adjustment, including partial scan extraction, coarse scan alignment, and loop closure detection. Then we illustrate our bundle adjustment algorithm in Sec. 5.6. Finally, we show results in Sec. 6.4.1.

5.1 Other Related Work

Tong et al. (2012) illustrated a full body scanning system with three Kinects. Their system uses a turntable to turn people around, but cannot handle large deformations. Other high-end multi-camera setups include (De Aguiar et al., 2008; Vlasic et al., 2008; Gall et al., 2009; Vlasic et al., 2009). In our work we wish to move away from complex rigs, and support more lightweight and commodity consumer setups, using only a single off-the-shelf depth sensor.

More lightweight capture setups have been demonstrated, but either still require complex lighting, more than one camera, or cannot generate high quality results (Hernández et al., 2007; Liao et al., 2009; Li et al., 2009; Weise et al., 2011; Valgaerts et al., 2012; Ye et al., 2012; Wu et al., 2013).

More severe deformations can be handled with template-based systems. For example, Zollhofer *et al.* (Zollhöfer et al., 2014) first acquire a template of the scene under near-rigid motion using Kinect fusion, and then adapt that template to nonrigid sequences. Even more specialized are systems based on human shape models (Vlasic et al., 2008; Weiss et al., 2011; Zhang et al., 2014). The shape prior means they cannot scan general shapes, including even humans holding objects, or in unusual clothing. More general approaches either work on diverse (non-rigged) templates (Hernández et al., 2007; De Aguiar et al., 2008; Liao et al., 2009; Li et al., 2009), or use template-less spatio-temporal representations (Mitra et al., 2007; Wand et al., 2009; Tevs et al., 2012). Instead our system discovers the *latent* surface model without the need for an initial rigid scan or statically captured template model. It also attempts to mitigate the drift inherent in non-template-based models.

5.2 Triangular Mesh Surface Model

Throughout this paper, we use a triangular mesh as our fundamental surface representation. We parameterize a triangle mesh by the set of 3D vertex locations $\mathcal{V} = \{\mathbf{v}_{m=1}^{M}\}$ and the set of triangle indices $\mathcal{T} \subset \{(i,j,k) : 1 \leq i,j,k \leq M\}$. We will also occasionally query the triangulation through the function $\mathcal{N}(m)$ which returns the indices of the vertices neighboring vertex m, or through the use of a variable $\tau \in \mathcal{T}$ representing a single triangle face.

We will often need to label a mesh using a subscript (e.g. $\mathcal{V}_i$) in which case we will label the vertices with a corresponding superscript (*e.g.* $\mathbf{v}_m^i$). Indeed, a point on the surface itself is parameterized using a surface coordinate $\mathbf{u} = (\tau, u, v)$ where $\tau \in \mathcal{T}$ is a triangle index and (u, v) is a barycentric coordinate in the unit triangle. The position of this coordinate can then be evaluated using a linear combination of the vertices in τ as

$$S(\mathbf{u}; \mathcal{V}) = u\mathbf{v}_{\tau_1} + v\mathbf{v}_{\tau_2} + (1-u-v)\mathbf{v}_{\tau_3} \tag{5.1}$$

and its surface normal computed as

$$S^{\perp}(\mathbf{u}; \mathcal{V}) = \frac{(\mathbf{v}_{\tau_2} - \mathbf{v}_{\tau_1}) \times (\mathbf{v}_{\tau_3} - \mathbf{v}_{\tau_1})}{\|(\mathbf{v}_{\tau_2} - \mathbf{v}_{\tau_1}) \times (\mathbf{v}_{\tau_3} - \mathbf{v}_{\tau_1})\|} \,. \tag{5.2}$$

5.3 Extracting Partial Scans

The first phase of our algorithm begins by preprocessing an RGBD sequence into a set of high quality, but only partial, scans $\{\mathcal{V}_i\}_{i=1}^{N}$ of the object of interest. Each of these segments is

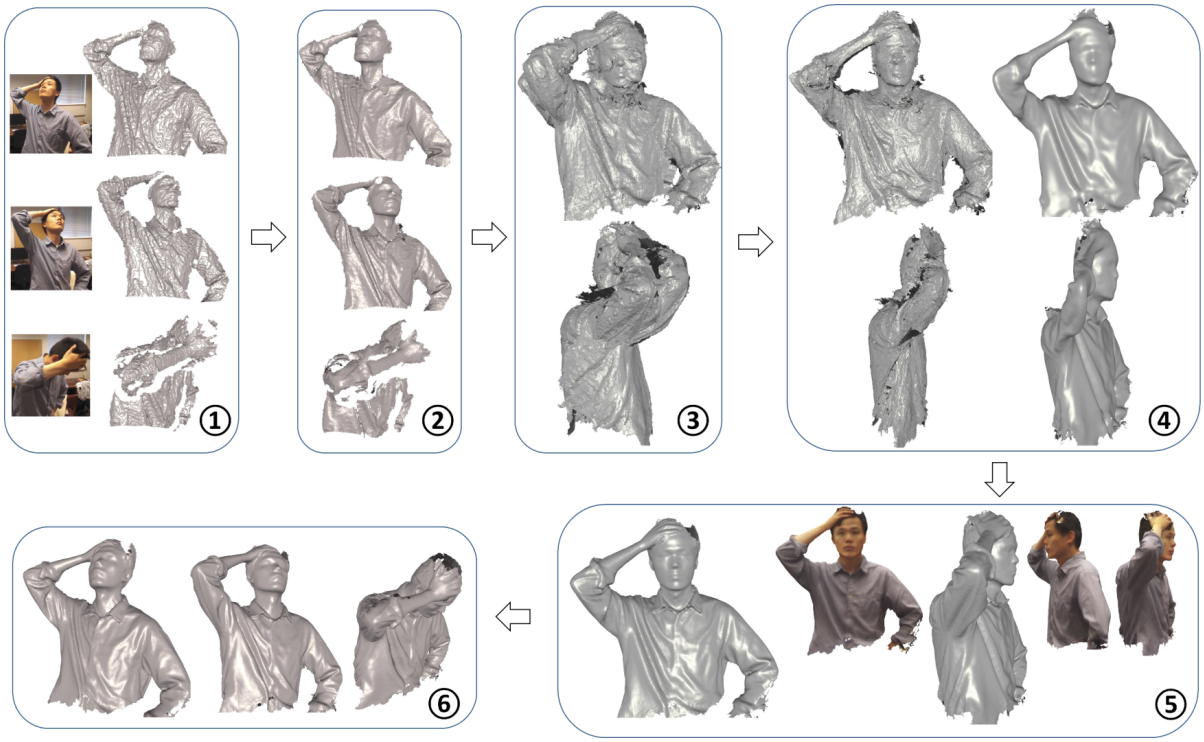


Figure 5.2: Scanning pipeline: ① input depth and color; ② partial scans; ③ coarse-aligned scans; ④(Left) aligned scans after Loop-Closure; ④(Right) Fused mesh after Loop-Closure; ⑤ mesh after bundle adjustment; ⑥ meshes deformed to every frame. The input sequence has around 400 frames which are fused into 40 partial scans (Sec. 5.3). Partial scans are consecutively placed in the reference pose to achieve the coarse alignment (Sec. 5.4). Next, loop closures are detected and alignment is refined (Sec. 5.5); all the LC-aligned scans are fused volumetrically to get the LC-fused surface which serves as the initial for the following bundle adjustment stage (Sec. 5.6). As a by-product of the system, the reconstructed model can be deformed back to each frame.

reconstructed from a small contiguous set of F frames by using the method introduced in the previous chapter to fuse the depth data into a triangular mesh. These short segments can be reliably reconstructed using standard methods, in contrast to larger sequences where camera and reconstruction drift generally leaves gross errors at loop closure boundaries. In addition, these segments compress the information contained in the full sequence, drastically reducing the computational complexity of fitting our surface model to the entire sequence as described in following sections.

To reconstruct the partial scan for segment i, we begin by iteratively fusing data from each frame $f \in \{1, ..., F\}$ into the reference frame which is set as the first frame. This is trivially accomplished for frame 1, so for frame $f \in \{2, ..., F\}$ we extract from the current volumetric representation of the reference frame, the reference mesh $\mathcal{V}_i^1$ and align it to frame f using an ED deformation with parameters G_i^f. Note that the parameters G_i^{f-1} can be used to initialize this optimization. We then observe the deformed mesh $ED(\mathcal{V}_i^1; G_i^f)$, and find a set of nearby points on $\mathcal{V}_i^f$ to establish a set of correspondences between $\mathcal{V}_i^f$ and $\mathcal{V}_i^1$. These correspondences can then be used to estimate a parameter set $\hat{G}_i^f$ that aligns $\mathcal{V}_i^f$ back to $\mathcal{V}_i^1$ in the reference frame (Sumner et al., 2007) and that can be used to volumetrically fuse the data from frame f into the reference frame (where $\mathcal{V}_i^1$ lives). After completing this operation for all frames, a single surface $\mathcal{V}_i$ is extracted from the volumetric representation using marching cubes.

After this initial fusing, we have obtained a set of partially reconstructed segments $\{\mathcal{V}_i\}_{i=1}^N$, each of which is a partial scan of the object of interest at a different time and in a different pose. Examples of partial scans are shown in Figure 5.2(b). Ultimately, we want all segments $\{\mathcal{V}_i\}_{i=1}^N$ to be explained by a single complete mesh $\mathcal{V}$ (we call it the **latent mesh**) and a set of ED graphs $\{G_i\}_{i=1}^N$ that deforms $\{\mathcal{V}_i\}_{i=1}^N$ to $\mathcal{V}$. But it is not immediately clear where to get such a mesh, and how to get a good initial estimate of the deformation parameters required to achieve this. Instead, we proceed by deforming each of these segments into the reference pose, fusing the results together into a complete mesh, and using the deformations to provide a good initial guess for the parameters that minimize an appropriate energy.

5.4 Coarse Scan Alignment

In this section, we describe how we find deformation parameters G_i for each segment $\mathcal{V}_i$ so that a set of roughly aligned meshes $\{ED(\mathcal{V}_i; G_i)\}_{i=1}^{N}$ can be obtained in the reference pose (*i.e.* pose of $\mathcal{V}_1$). We first align each segment $\mathcal{V}_i$ to its immediate neighbor $\mathcal{V}_{i+1}$ yielding a parameter set $G_{i \to i+1}$ by using the nonrigid alignment technique introduced in the previous chapter. This is straightforward as adjacent scans have similar poses and the $G_{i \to i+1}$ can be initialized using the parameters already estimated by (Dou et al., 2013) when aligning the first frame to the last frame of segment i.

To obtain an alignment of segment $\mathcal{V}_{i+1}$ back to the reference frame, it is helpful to assume that we have already obtained such an alignment for segment $\mathcal{V}_i$, which is trivial for $i = 1$. Then for each vertex $\mathbf{v}_m^i$ of mesh $\mathcal{V}_i$, we find the nearest surface point $\mathbf{v}_{\mu(m)}^{i+1}$ on $\mathcal{V}_{i+1}$ (closer than 1cm) to its deformed position $ED(\mathbf{v}_m^i, G_{i \to i+1})$. Similarly, the alignment parameter set G_i tells us that $\mathbf{v}_m^i$ should be located at $\tilde{\mathbf{v}}_m^i = ED(\mathbf{v}_m; G_i)$ in the reference frame. This process establishes a set of correspondences $\{\langle \mathbf{v}_{\mu(m)}^{i+1}, \tilde{\mathbf{v}}_m^i \rangle\}_{m=1}^{M}$ which provide constraints that can be used to estimate G_{i+1} using the standard ED alignment algorithm (Sumner et al., 2007).

5.5 Error Redistribution

Naturally, the error in the propagation step accumulates, making the deformation parameter sets more and more unreliable as i increases. On the other hand, we assume that our sequence includes a loop closure and thus there should be some later segments that could match reasonably well to earlier segments. We would thus like to identify such pairs and establish rough

constraints between them, in the form of correspondences, so that the deformations can be refined. To this end, we consider matching the aligned scan $ED(\mathcal{V}_i; G_i)$ against the aligned scans $\{ED(\mathcal{V}_j; G_j)\}_{j=1}^{i-K}$, where $K \geq 1$ restricts to frames with enough movement. To measure the overlap of a mesh $\mathcal{V}_j$ and a mesh $\mathcal{V}_i$, we define the overlap ratio

$$d(\mathcal{V}_i, \mathcal{V}_j) = \frac{1}{M_i} \sum_{m=1}^{M_i} I[\min_{m'} \|\mathbf{v}_m^i - \mathbf{v}_{m'}^j\| < \delta] \tag{5.3}$$

as the proportion of vertices in $\mathcal{V}_i$ that have a neighboring vertex in $\mathcal{V}_j$ within δ (we use δ =4cm). We thus calculate $d_j^i = d(ED(\mathcal{V}_i; G_i), ED(\mathcal{V}_j; G_j))$ and consider as possible candidates, the set of scan indices $\mathcal{J}_i = \{j : d_j^i \geq r_1, |i-j| > K, d_j^i > d_{j-1}^i, d_j^i > d_{j+1}^i\}$, the indices whose aligned scan is at least K indices away with a 'peak' overlap ratio of at least r_1. For any scan index $j \in \mathcal{J}_i$, we then consider doing a more expensive, but more accurate, direct alignment of $\mathcal{V}_j$ to $\mathcal{V}_i$ using a set of ED parameters $G_{j\to i}$ (Dou et al., 2013). If $d(\mathcal{V}_i, ED(\mathcal{V}_j, G_{ji})) \geq r_2$ we then find a set of correspondences $\mathcal{C}_{ij} \subseteq \{1, ..., M_i\} \times \{1, ..., M_j\}$ for which for any $(m, m') \in \mathcal{C}_{ij}$, we have that$\|\mathbf{v}_m^i - ED(\mathbf{v}_{m'}^j, G_{j\to i})\|$ is less than 1 cm. We set $\mathcal{C}_{ij} = \emptyset$ for any other pairs of frames that did not pass this test. In our experiment we let $r_1 = 30\%, r_2 = 50\%$.

With these loop closing correspondences extracted, we use Li *et al.*'s algorithm (Li et al., 2013) to re-estimate ED graph parameters $\mathcal{G} = \{G_i\}_{i=1}^N$, by minimizing the energy;

$$\min_{\mathcal{G}} \lambda_{\text{corr}} E_{\text{corr}}(\mathcal{G}) + \lambda_{\text{reg}} \sum_i E_{\text{reg}}(G_i) + \lambda_{\text{rigid}} \sum_i E_{\text{rigid}}(G_i), \tag{5.4}$$

where

$$E_{\text{corr}}(\mathcal{G}) = \sum_{\substack{i=1 \\ j=1 \\ j \neq i}}^{N} \sum_{(m,m') \in \mathcal{C}_{ij}} \| ED(\mathbf{v}_m^i; G_i) - ED(\mathbf{v}_{m'}^j; G_j) \|^2 \,. \tag{5.5}$$

After the set of deformation parameters $\mathcal{G}$ is estimated, we deform the scans accordingly and fuse them volumetrically to obtain a rough latent surface $\mathcal{V}$. Fig. 5.2(b,c)&5.3(b) show examples of scan alignment before and after loop closure.

5.6 Dense Nonrigid Bundle Adjustment

At this point, the above procedure has succeeded in giving us a rough surface representation of our object of interest, but the process has washed out the fine details that can be seen in the partial scans (see Fig. 5.2 and Fig. 5.11). This is largely a result of the commitment to a set of noisy correspondences used for error distribution. Eq. 5.4 does not aim to refine these correspondences, and thus misalignments are inevitable. As shown in Fig. 5.2 where large deformation exists, the misalignment is still visible where a loop closure has occurred, and the fused model looks flat and misses many details.

To improve both the data alignment and recover the fine details we employ a bundle adjustment (BA) type technique to refine $\mathcal{V}$ as to explain all the data summarized in the partial scans $\{\mathcal{V}_i\}_{i=1}^N$. We parameterize the deformation that each partial scan $\mathcal{V}_i$ has to undergo to be explained by the reference $\mathcal{V}$ using a set of ED deformation parameters G_i. We then cast an energy $E(\mathcal{V})$ over the latent mesh $\mathcal{V}$ as a combination of the following terms.

5.6.1 Deformation Terms

For each data point $\mathbf{v}_m^i$ in segment $\mathcal{V}_i$, we expect that some ED graph G_i deforms it towards the latent mesh $\mathcal{V}$ and $ED(\mathbf{v}_m^i; G_i)$ gets explained by $\mathcal{V}$. We thus add an energy term designed to encourage the distance of $ED(\mathbf{v}_m^i; G_i)$ to the latent surface to be close, and for the normal to match. This term is

$$\begin{aligned} E_{\text{data}}(\mathcal{V}) = \sum_{i=1}^{N} \min_{G_i} \sum_{m=1}^{M_i} \min_{\mathbf{u}} \; & \lambda_{\text{data}} E_{\text{point}}(\mathbf{v}_m^i; G_i, \mathbf{u}, \mathcal{V}) + \\ & \lambda_{\text{normal}} E_{\text{normal}}(\mathbf{n}_m^i; G_i, \mathbf{u}, \mathcal{V}) \\ & + \lambda_{\text{reg}} E_{\text{reg}}(G_i) + \lambda_{\text{rigid}} E_{\text{rigid}}(G_i) \end{aligned} \tag{5.6}$$

where

$$E_{\text{point}}(\mathbf{v}; G, \mathbf{u}, \mathcal{V}) = \|ED(\mathbf{v}; G) - S(\mathbf{u}; \mathcal{V})\|^2 \tag{5.7}$$

and

$$E_{\text{normal}}(\mathbf{n}; G, \mathbf{u}, \mathcal{V}) = \|ED(\mathbf{n}; G) - S^{\perp}(\mathbf{u}; \mathcal{V})\|^2 \,. \tag{5.8}$$

$S(\mathbf{u}; \mathcal{V})$ and $S^{\perp}(\mathbf{u}; \mathcal{V})$ are corresponding point and normal of $ED(\mathbf{v}; G)$[1] in the latent surface $\mathcal{V}$, which we have explained in Section 5.2.

As we continue to use the ED deformation model, the terms $E_{\text{reg}}(G)$ and $E_{\text{rigid}}(G)$ continue to provide regularization for ED graphs.

[1]Note that we do not set ED graph G_i on the latent mesh $\mathcal{V}$ to deform $\mathcal{V}$ towards partial scan $\mathcal{V}_i$ and minimize $\sum_{i=1}^{M_i} \min_{\mathbf{u}} \|\mathbf{v}_m^i - S(\mathbf{u}; ED(\mathcal{V}; G))\|^2$, because this gives many unnecessary ED nodes given that $\mathcal{V}$ is complete and $\mathcal{V}_i$ is partial.

5.6.2 Surface Regularization Terms

In addition, we regularize the latent mesh using the Laplacian regularizer

$$E_{\text{lap}}(\mathcal{V}) = \sum_{m=1}^{M} \|\mathbf{v}_m - \frac{1}{|\mathcal{N}(m)|} \sum_{m' \in \mathcal{N}(m)} \mathbf{v}_{m'}\|^2 \,, \tag{5.9}$$

where $\mathcal{N}(m)$ is the set of indices of vertices that neighbor $\mathbf{v}_m$. This term attracts a vertex to the centroid of its neighbors, penalizing unevenness of the surface, but has the potential to shrink the surface by dragging the set of boundary vertices inwards. We thus also add a energy term encouraging isometry as

$$E_{\text{iso}}(\mathcal{V}) = \sum_{m \in \mathcal{B}} \sum_{m' \in \mathcal{N}(i)} |\|\mathbf{v}_{m'} - \mathbf{v}_m\|^2 - L_{mm'}^2|^2 \tag{5.10}$$

where $\mathcal{B} \subseteq \{1, ..., M\}$ is the set of indices of such boundary vertices, and $L_{mm'}$ is the length $\|\mathbf{v}_{m'} - \mathbf{v}_m\|$ in the initial mesh.

5.6.3 Solving

Combining all of the above energy terms (Eq. 5.6, 5.9, and 5.10), we obtain the full energy

$$E(\mathcal{V}) = E_{\text{data}}(\mathcal{V}) + \lambda_{\text{lap}} E_{\text{lap}}(\mathcal{V}) + \lambda_{\text{iso}} E_{\text{iso}}(\mathcal{V}) \tag{5.11}$$

that we seek to minimize. To deal with the inner minimizations, we follow the lead of (Taylor et al., 2014; Zollhöfer et al., 2014), defining a set of latent variables, passing them through

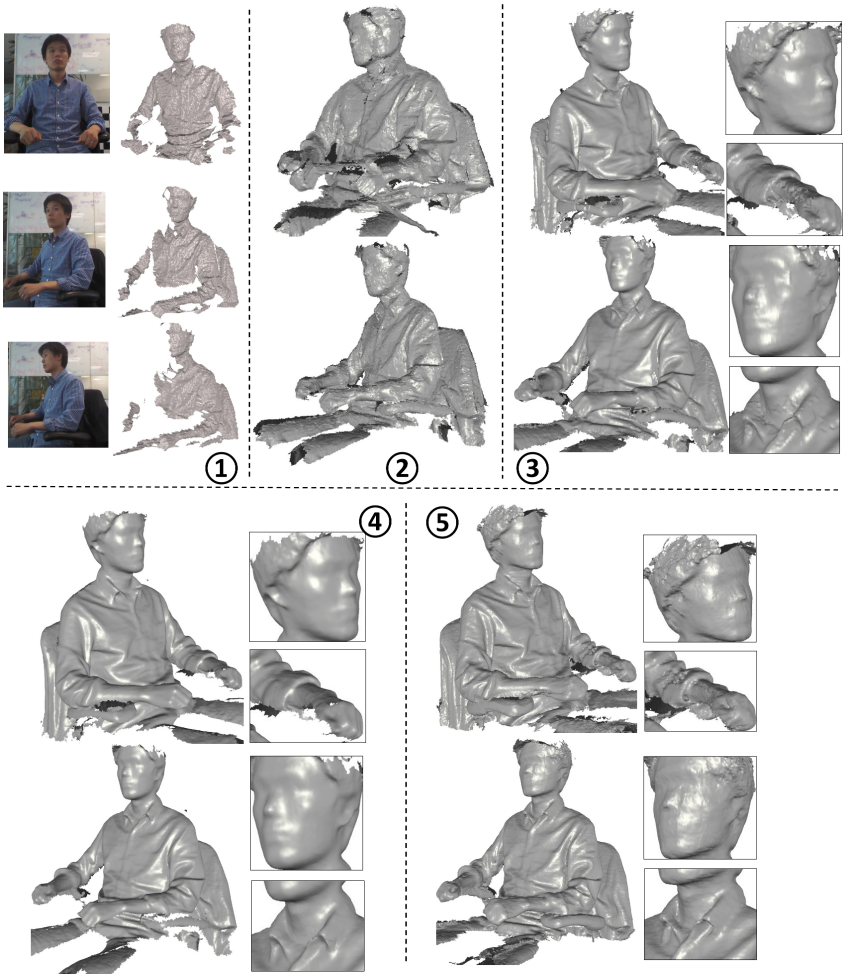


Figure 5.3: Scanning a person with slight deformation: ① input depth and color; ② before/after LC; ③ LC-fused surface; ④ BA-optimized surface; ⑤ KinectFusion results. Before loop closure (LC), scans are poorly aligned. After LC, the surface is topologically correct but noisy. Bundle Adjustment (BA) removes spurious noise without further smoothing details such as the shirt collar.

the sums, and rewriting the energy in terms of a lifted energy defined over these additional latent variables. In our case, we have the ED deformation parameter sets $\mathcal{G} = \{G_i\}_{i=1}^{N}$ and the surface coordinates $\mathcal{U} = \{\mathbf{u}_1^m\}_{m=1}^{M_1} \cup ... \cup \{\mathbf{u}_N^m\}_{m=1}^{M_N}$, which allows us to obtain a lifted energy $E'(\mathcal{V}, \mathcal{G}, \mathcal{U})$ such that

$$E(\mathcal{V}) = \min_{\mathcal{G},\mathcal{U}} E'(\mathcal{V}, \mathcal{G}, \mathcal{U}) \leq E'(\mathcal{V}, \mathcal{G}', \mathcal{U}') \tag{5.12}$$

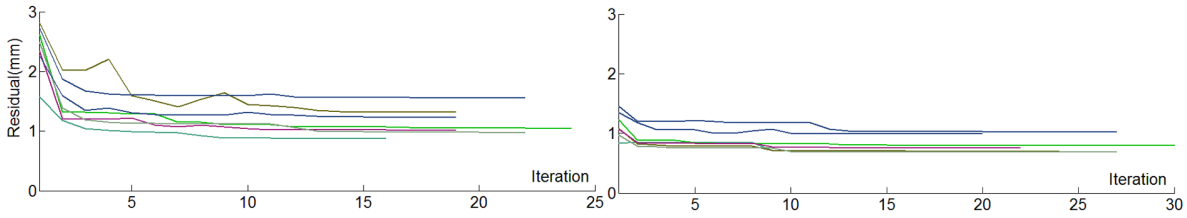


Figure 5.4: Partial scan alignment residuals during bundle adjustment. Left: Low Resolution BA; Right: High Resolution BA.

for any $\mathcal{G}'$ and $\mathcal{U}'$. We can thus minimize our desired energy by minimizing this lifted energy and to this end, we notice that all terms are in a sum of squares form. We thus use the Levenberg–Marquardt algorithm implemented in Ceres (Agarwal and Mierle, 2013) to minimize $E'(\mathcal{V}, \mathcal{G}, \mathcal{U})$. We initialize the latent mesh $\mathcal{V}$ using the coarse mesh recovered in the previous section, and $\mathcal{G}$ using the corresponding ED parameter sets and $\mathcal{U}$ by conducting a single closest point computation.

Note that even though surface normal $S^{\perp}(\mathbf{u}; \cdot)$ is constant with respect to the barycentric coordinate $\mathbf{u}$ (an entire triangle on the latent surface shares the same normal vector), it does give constraints to the latent mesh and the ED graphs, which makes latent surface smooth and improves the alignment.

Note that some special care has to be taken to allow the Levenberg-Marquardt algorithm to interact with a surface coordinate variable $u \in \mathcal{U}$ (Taylor et al., 2014; Cashman and Fitzgibbon, 2013). Such a variable has the atypical parameterization $\mathbf{u} = (\tau, u, v)$ where τ is discrete (a triangle ID), and (u, v) are real valued coordinates in the unit triangle. As typically the coordinate (u, v) will lie strictly within the unit triangle, τ remains constant locally and only the Jacobians with respect to (u, v) which are well defined are provided to the optimizer. When an update $(u, v) \leftarrow (u, v) + \delta(du, dv)$ is requested that would exit the unit triangle, the coordinate

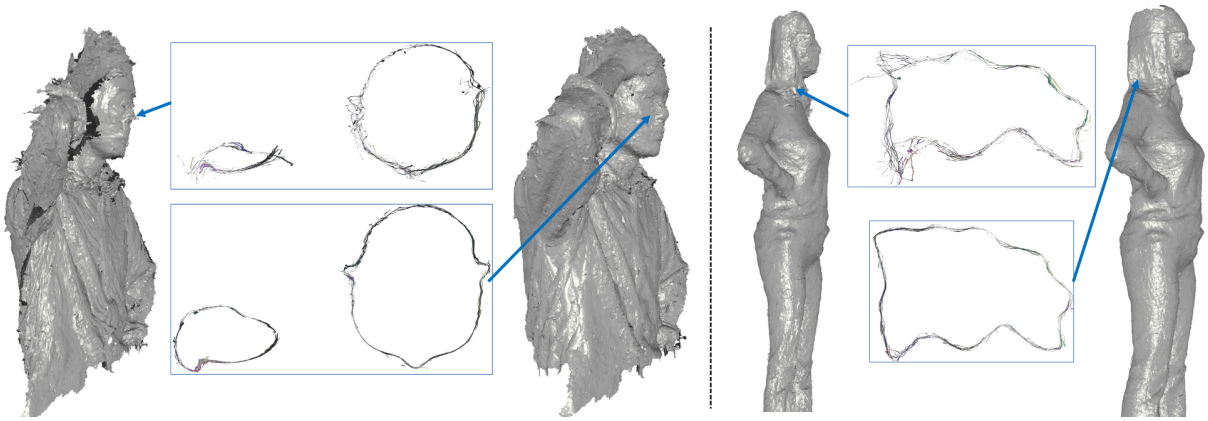


Figure 5.5: Two examples of aligned scan before and after BA (Left: before; Right: after). The cross sections of scans are given in the middle.

should first move the distance $\hat{\delta}$ to the edge of the triangle. The adjacent triangle τ' is then looked up, a new direction (du', dv') and step size $\delta' = \delta - \hat{\delta}$ computed, and finally the procedure is recursively called after updating $\tau \leftarrow \tau'$, $(du, dv) \leftarrow (du', dv')$ and $\delta \leftarrow \delta'$. Eventually the step size δ will be sufficiently small that an update does not need to leave a triangle.

5.7 Experiments

5.7.1 Implementation details

In the following experiments, we evaluate our method on a variety of RGBD sequences of various objects of interest. Each sequence is between 200 and 400 frames, and we fuse these volumetrically into 20 to 40 partial scans by fusing the data from each $F = 10$ frame subsegment. We set the size of the voxels in the fusion procedure to 2mm cubed when scanning a close object and 3mm cubed for objects at a further distance. This results in partial scans with around 100,000 vertices. When conducting nonrigid alignment for both partial scan extraction and alignment, ED nodes are sampled as to remain roughly at 5cm (measured in

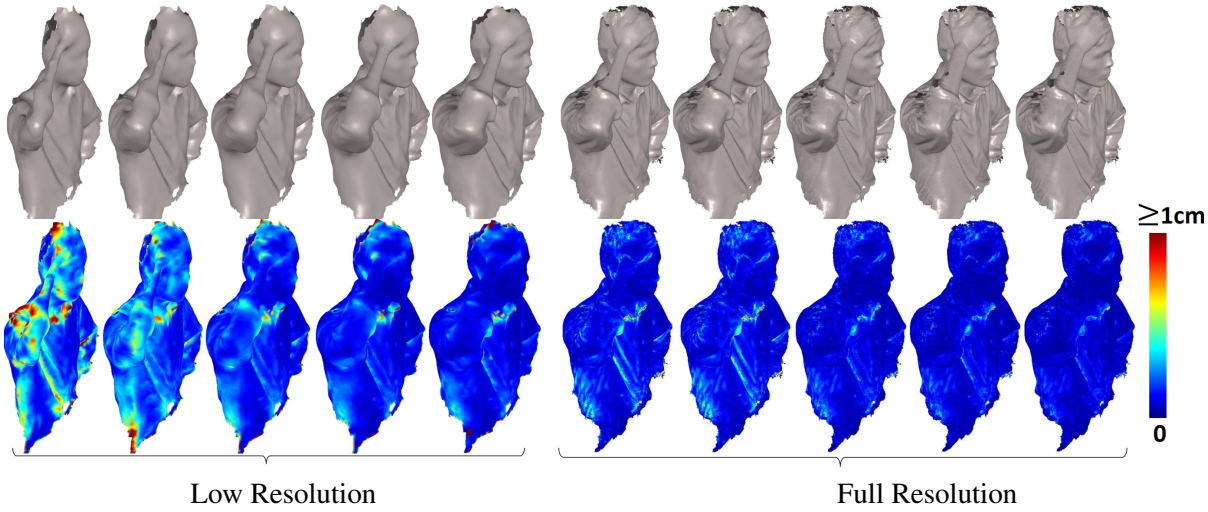


Figure 5.6: Bundle adjustment iterations. Top row: evolving surface model. Bottom: per-vertex residuals. Note the increase in detail of the right forearm and hand.

geodesic distance) to their neighbors. This endows each ED graph with roughly 150 to 200 nodes depending on the dimension of the object of interest.

After detecting the loop closure constraints and performing error redistribution, the aligned partial scans are volumetrically fused to get an initial latent mesh for the final bundle adjustment (BA) stage. We perform bilaterial filtering on the volume data to ameliorate any misalignment. We also perform a simple remeshing to eliminate thin triangles on initial latent mesh extracted with marching cubes, which makes BA numerically stable.

The bundle adjustment is the most expensive stage, given the huge amount of parameters to be optimized in Eq. 5.12: the roughly 5,000 graph nodes, 300,000 vertices of the latent mesh and three million surface coordinates. A limitation of this procedure is that the number of vertices on the latent mesh and its triangulation remain fixed throughout the bundle adjustment stage. Thus, if the initial mesh does not have the correct shape topology or has missing parts due to poor initial alignment, it is difficult for the bundle adjustment to recover the correct

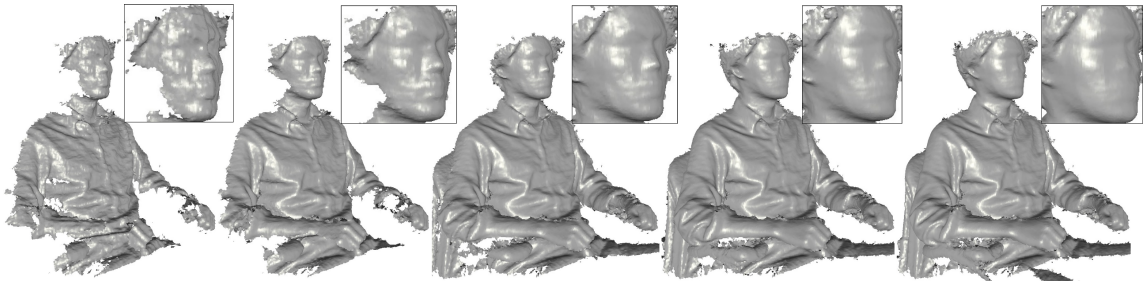


Figure 5.7: KinectFusion with nonrigid alignment. The accumulated surfaces after fusing 10, 30, 50, 70, 90 frames are shown. Note the nose gets blurred at the end.

shape. To handle the above issues, we take a coarse-to-fine approach by running the bundle adjustment twice with different levels of detail.

In the first run, a low resolution latent mesh is used with an average distance between neighboring vertices of 1cm. The first run quickly converges and improves partial scan alignments $\mathcal{G}$ significantly, from which a better initial latent mesh can be built. In the second run, we use the full resolution mesh where the average distance between neighboring vertices is about 2mm. Initializing the parameters from the previous bundle adjustment, the vertices on the latent mesh do not need to move much along the tangent direction, so we constrain the vertex to only move as a displacement along the direction normal to the initial latent mesh, which reduces the number of parameters on the latent surface by nearly two thirds. That is, only one single displacement parameter per vertex instead of three is required to parameterize full 3D position.

Fig. 5.6 illustrates the intermediate latent surfaces together with the alignment residual at each BA iteration; the alignment error is computed for each vertex on the latent surface as its average distance to the deformed partial scans. Fig. 5.4 plots the average alignment residuals during BA (including both accepted and rejected BA iterations) on various data sequences. The

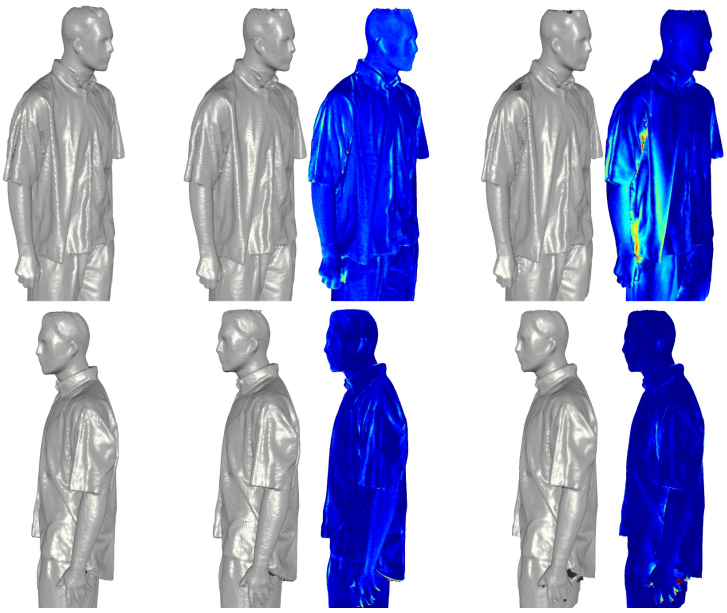


Figure 5.8: Static Scanning. Error maps use the same scale as Fig. 5.6.

alignment error typically goes down from 3mm to less than 1mm. Examples of aligned scans before and after BA are also given in Fig. 5.5, where the cross sections of scans are shown to demonstrate the alignment quality and the bundle adjustment's ability to recover the true structure of the object.

5.7.2 Comparison with KinectFusion

Our system is designed for dynamically moving objects, but it still works in more restricted cases such as rigid scenes (*i.e.* scanning static objects). in reconstruction quality between our method and KinectFusion on a static mannequin. To compare the two systems quantitatively, we first generate a 3D model of the mannequin which serves as the ground truth and then synthesize a sequence of depth maps and color images by moving a virtual camera around the 3D model. We run our algorithm and KinectFusion on the synthetic data. As shown in Fig. 5.8,

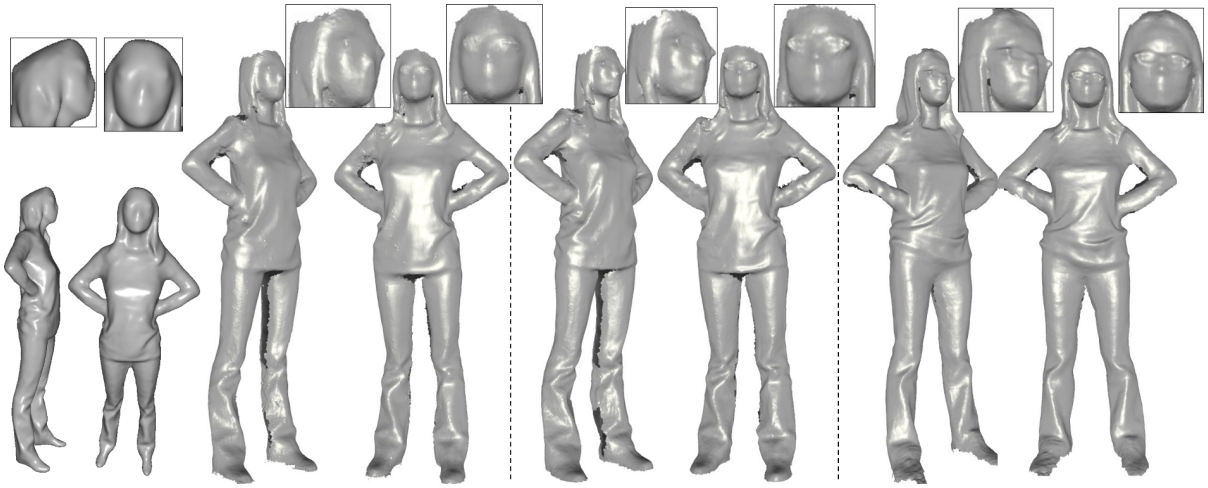


Figure 5.9: Comparison with 3D Self-Portraits. Scanning results of (a) shapifyme; (b) 3D self-portraits implemented by us; (c) BA-optimized 3D self-portraits; (d) our system.

both systems give appealing reconstructions which are authentic to ground truth. KinectFusion has an average reconstruction error of 0.94 mm v.s. 1.21 mm in our system. Our system has lower residual on the side that is observed by the reference frame (1st row in Fig. 5.8, error map uses the same scale as Fig. 5.6) while has higher residual on the other side (2nd row in Fig. 5.8) due to flexibility introduced by the nonrigid alignment. Naturally, we don't expect to outperform a method that exploits the rigidity of this scene, but we are satisfied that our system can get similar results without requiring such assumptions.

In contrast though, KinectFusion fails in dynamic cases. Fig. 5.3 shows the reconstruction results of KinectFusion on a sequence with slight head movement. Replacing ICP in KinectFusion with the nonrigid alignment algorithm (Dou et al., 2013) does not result in a reasonable reconstruction either. As shown in Figure 5.7, when non-rigidly fusing more than 30 frames, the drifting artifacts result in a blurred nose.

5.7.3 Comparison with 3D Self-portraits

3D self-portraits (Li et al., 2013) is among the first systems with the capability to scan a dynamic object with a single consumer sensor. We want to stress that our system handles continuously deforming objects while 3D self-portraits first reconstructs eight static scans and then non-rigidly fuses them later. The above difference prevents us from comparing two system quantitatively, but we show side-by-side of the reconstructed models of the same person from the two systems in Fig. 5.9. The software Shapifyme which implements 3D self-portraits appears to heavily smooth the reconstructions, and our implementation of 3D self-portrait gives more detailed reconstructions. We then ran bundle adjustment algorithm of Sec. 5.6 on the eight scans, and we find that it improves the reconstruction further, showing another advantage of our approach. Compared with 3D self-portraits, our system allows continuous movement and recovers more facial details.

5.7.4 Synthetic sequence

We tested our system on the Saskia dataset (Vlasic et al., 2009) which contains dramatic deformations. The original sequence has a roughly complete model at each frame, and thus we synthesize one depth map and color image from each frame with a virtual camera rotating around the subject. Our reconstruction system results in a shape in a reference pose (*i.e.* the latent mesh $\mathcal{V}$) as shown on the left of Fig. 5.10. To measure alignment error, we then deform $\mathcal{V}$ to each frame and compute the distance from the frame data. To achieve this a backward ED graph $\tilde{G}_i$ from $\mathcal{V}$ to each partial scan $\mathcal{V}_i$ is first computed using correspondences. The

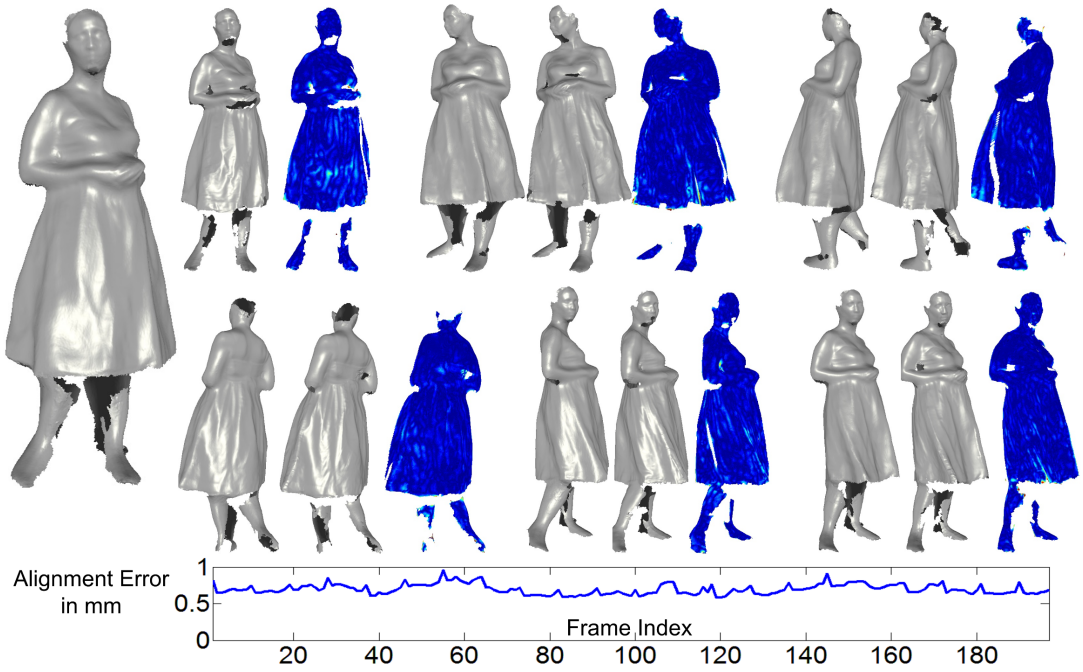


Figure 5.10: Alignment error in Saskia dataset. The first shape in each triple is the deformed reconstructed surface, the second is the ground truth, and the third shows the alignment error (same scale as Fig. 5.6). Per-frame alignment error is drawn at the bottom.

deformations from partial scan $\mathcal{V}_i$ to the frames in the segment i have already computed as explained in Section 5.3, so we first deform $\mathcal{V}$ to each partial scan's pose and then to each frame's pose. The alignment error is then measured between the deformed reconstruction and the synthesized depth map. We draw the alignment error at each frame at the bottom of Fig. 5.10.

The Saskia sequence poses a particular challenge as the topology changes when the dress touches the legs. This introduces some artifacts on the legs (*e.g.*, expansions) in the reconstructed latent mesh $\mathcal{V}$ and also gives some problems in the deformed latent mesh in each frame's pose.

5.7.5 Scanned example

Fig. 5.3 shows a sequence with small deformations. The loop closure technique described in Sec. 5.5 reconstructs a reasonable model, but some artifacts exist due to misalignment. Our bundle adjustment technique in Sec. 5.6, however, improves the reconstruction. Another example with considerable deformations is shown in Fig. 5.2, where the loop closure gives a problematic alignment of the partial scans and a poor reconstruction (*e.g.* the arm is unrealistically thin). During bundle adjustment, the arm gradually expands as optimization iterations are performed until it is a realistic size (see Fig. 5.6).

We tested our system on several situations including full body scans and upper body scans. We also tried to scan objects other than human beings. Fig. 5.11 shows some scan examples. In all the scans that we performed, the Kinect sensor is mounted on a tripod, and we let people turn around freely in front or, in the case of an object, be rotated by the "director" of the scene.

5.8 Discussions

We have presented a system which merges a sequence of images from a single range sensor into a unified 3D model, without requiring an initial template. In contrast to previous systems, a wider range of deformations can be handled, including wriggling children. Some limitations remain, however. First, although complex scene topologies can be handled, the topology is restricted to be constant throughout the sequence, and if the coarse-scale reconstruction does not correctly choose the topology, it cannot currently change at the fine scale.

The computational cost is also high. We run our experiments on a desktop PC with 8-

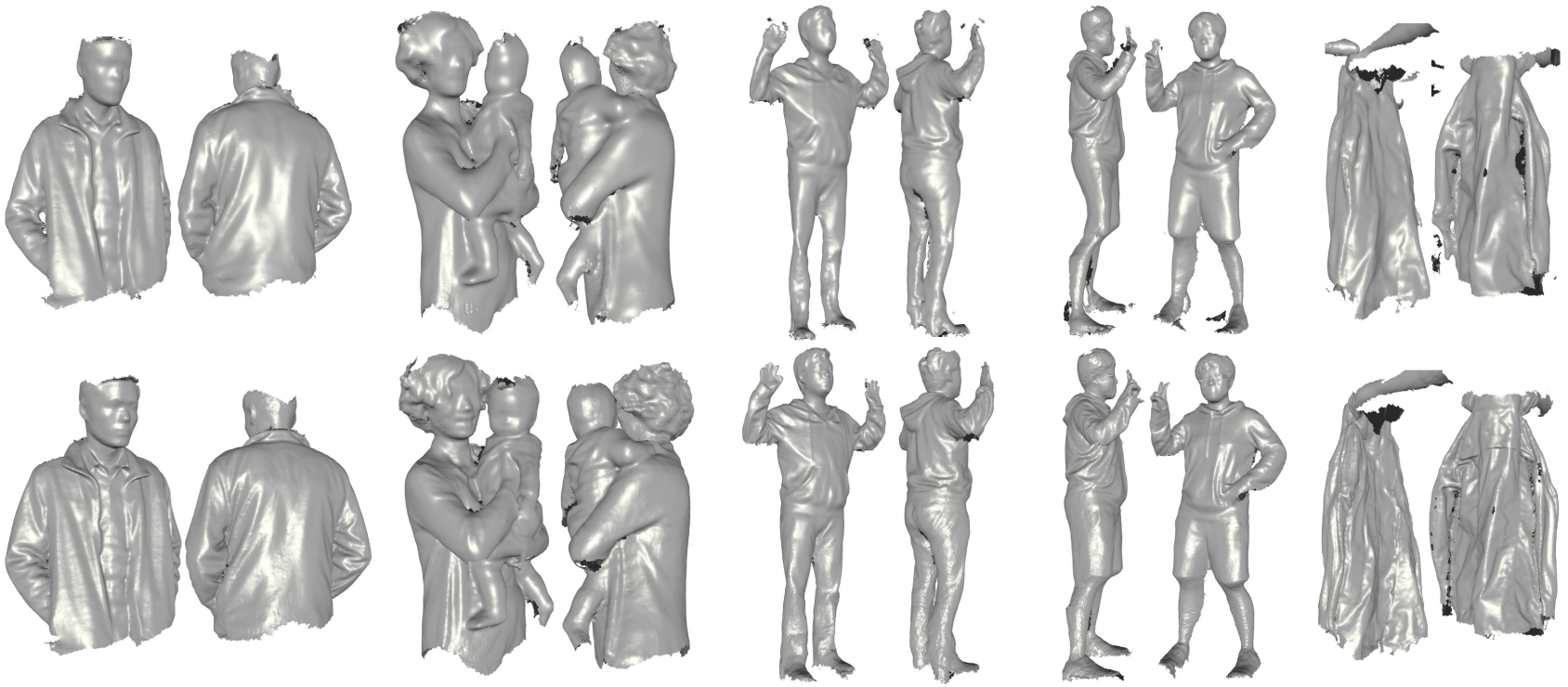


Figure 5.11: Top: reconstruction after loop closure. Bottom: final reconstruction after bundle adjustment.

core 3.0G Hz Intel Xeon CPU and 64G memory. For a sequence with 400 frames, and the partial scan preprocessing stage takes around 30 seconds per frame, initial alignment and loop closure detection takes about 1 hour, and final bundle adjustment up to 5 hours. However, these results are using only lightly optimized implementations, and if we were to assume that one use of these models would be to 3D print a "shelfie", the 3D printing process will itself take a considerable time. Even if the goal is to upload the model to a game, an overnight process remains valuable.

CHAPTER 6: Room-sized Dynamic Scene Reconstruction

In this chapter, we demonstrate a system that is capable of capturing complete 3D structures of a room-sized dynamic scene, including furniture and its inhabits (human beings). We achieve this by mounting multiple units of depth sensors on the walls and incorporating temporal information to different degrees for different parts of the scene. We divide the indoor environment into following three categories:

1. Static Background: objects that do not move or deform, such as the room frame (i.e. walls, floor, and ceiling) and sturdy furniture (e.g. a large sofa).
2. Semi-Static Objects: objects that do not deform and move infrequently, such as small furniture (e.g. chairs and tables) and decorating objects or appliances (e.g. teapots or cups).
3. Dynamic Objects: objects that continuously move and deform, such as human beings.

For the dynamic objects, a fixed depth camera mounted on the wall can observe different perspective of the objects at different times since they move. Since these perspectives complement each other, accumulating the 3D data over time can fill up holes and filter out noise as shown in previous chapters. For static and semi-static objects, a fixed camera almost always views the same perspectives; thus the 3D data quality does not improve while accumulating data. Instead, we first scan them off-line by moving one depth camera over the room and then segment the semi-static models (e.g., furnitures) from the static background. During the on-line operation,

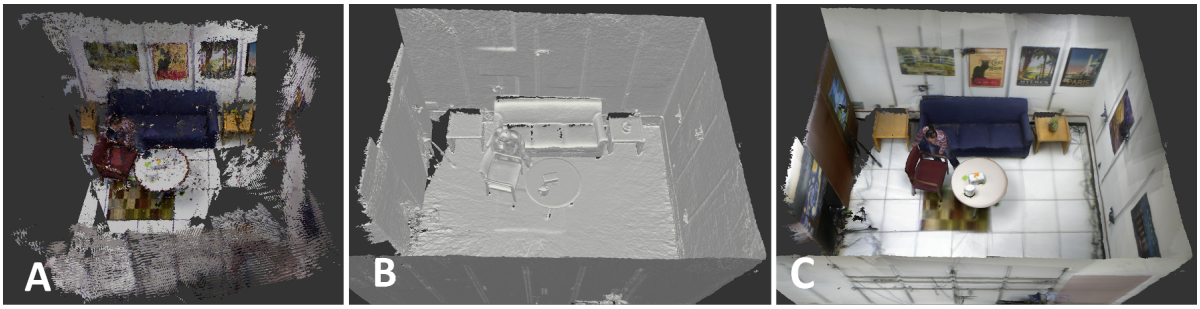


Figure 6.1: Room-sized Dynamic Scene Reconstruction. (A) Originally captured point cloud from 10 Kinects; (B) Enhanced result of our system; (C) Enhanced and textured result of our system.

we replace the observed live static background with the pre-scanned static background. Similarly, the pose of a semi-static object is first estimated and its pre-scanned model is transformed accordingly to replace the live data.

The core of the system is a unified surface tracking algorithm that aligns previous accumulated dynamic models and pre-scanned semi-static models with the live observations at one frame. The algorithm allows for close interactions between objects, such as a person sitting on a chair, and topology changes of dynamic objects. We will discuss our system implementation in Section 6.2.

We placed 10 Kinects on the walls in a room and captured various sequence to test our system. Section 6.4 gives implementation details and experimental results. The tracking algorithm in the system is computationally expensive due to the high dimensionality of the parameter space, so our current CPU implementation does not run in real-time yet, while we expect parallelized computation with GPU would dramatically boost the performance.

Figure 6.2: Scanned Room Model and Scene Segmentation. The left shows the original scanned model; the right shows various segmented parts. The tea pot and the tissue box on the table is also manually segmented out.

6.1 Related work

Maimone and Fuchs (Maimone and Fuchs, 2011, 2012) have designed various 3D capture systems using multiple Kinects. Their vibration system reduced the interference problem between Kinects; but the Kinects' limited accuracy on far objects and occlusions between objects limited the performance. Beck et.al. (Beck et al., 2013) demonstrates their 3D immersive telepresence system with a room-sized scene capture, but they did not improve the quality of the 3D capture using temporal information either.

Use temporal information for improved 3D structure is not a new idea; KinectFusion (Izadi et al., 2011; Newcombe et al., 2011) shows its success of fusing depth maps to a complete model for static scene with one moving Kinect. Our system instead uses multiple fixed Kinects to capture a dynamic scene. Our system introduced in previous chapters is able to scan and track a single dynamic object with a nonrigid matching algorithm. In contrast, this work aims at a room-sized capture system that deals with multiple dynamic objects and static objects. In

addition, our system allows some degree of topology changes during data accumulation for dynamic objects.

Our work also relates to Li's work on shape completion (Li et al., 2012) and surface tracking (Bojsen-Hansen et al., 2011). In the former work, holes on a single person in a frame is filled with the information from adjacent frames via nonrigid matching, and the latter work tracks surfaces with topology changes. Our system only uses the information available before the current frame; thus it is suitable for on-line operation (neither Li's system and our system runs in real time). In addition, we deal with topology variation with a simpler two-staged matching algorithm.

6.2 Room-sized Dynamic Scene Reconstruction

As aforementioned, the key insight of our system is to partition an indoor scene into three categories (static, semi-static, and dynamic) and treat each category differently when incorporating temporal information. Our system has two stages: off-line scanning and on-line running. We scan the static parts (static background and semi-static objects) of the room off-line and segment the scanned model into isolated triangles meshes, such as room frames and furnitures. Each triangle mesh is labeled manually as either static or semi-static.

The prescan of the room is aligned with the live data (depth maps and color images) from 10 Kinects beforehand as explained in Section 6.3. During the on-line operation, the scanned model of static background directly replaces its counterpart in the live data. We track the rigid movements of the semi-static objects and transform the scanned models accordingly. We also track the nonrigid movements of dynamic objects and accumulate their depth data for

complete and noise-free models. To deal with the interaction between objects, we use a unified framework to track both semi-static objects and dynamic objects simultaneously, as introduced in Section 6.2.2. The on-line system pipeline is detailed in Section 6.2.3.

6.2.1 Room Scanning and Segmentation

We scan the static background and semi-static objects together by capturing a sequence of RGB-D frames with one moving Kinect. The plane and feature based bundle adjustment system introduced in Chapter 2 is used to align all frames together. A combination of a volumetric depth map fusion algorithm (Curless and Levoy, 1996) and the Marching Cubes algorithm (Lorensen and Cline, 1987) generates a triangle mesh of the room. The by-products of our room scanning system are the dominant planes in the room, which is used to extract the room frames. We detect six planes (correspond to the floor, walls and ceiling) composing of a box that contains most data points; the surface points within certain distance from one of these planes are denoted as room frame points. The remaining surface mesh after eliminating room frame represents everything inside the room. We run connected component labeling on this surface mesh to segment it into isolated triangle meshes. Each isolated triangle mesh is then manually identified as a semi-static or static object. With minor manual efforts, all the static and semi-static objects are segmented. Figure 6.2 shows the scanned model of a room and its segmented pieces: room frame, sofa, table, etc.

6.2.2 Unified Tracking Algorithm

The tracking algorithm introduced in this section transforms or deforms pre-scanned semi-static models and previously accumulated dynamic models to fit the observations at current frame under one unified framework. The accumulation of dynamic models will be explained later in Section 6.2.3, and let's assume they are ready for use at this point. From now on, we use $\{\mathcal{S}_i\}_{i=1}^{M}$ to denote pre-scanned semi-static models and $\{\mathcal{T}_j\}_{j=1}^{N}$ for accumulated dynamic models; and we use $\{\tilde{\mathcal{S}}_i\}_{i=1}^{M}$ and $\{\tilde{\mathcal{T}}_j\}_{j=1}^{N}$ for the transformed models to be aligned with the current observation. Surfaces $\mathcal{S}$ and $\mathcal{T}$ are represented by triangulated meshes. The observations at a frame include depth maps and color images.

Compared with tracking the movement of each object individually, the unified algorithm is able to deal with the close contact between objects. As the nonrigid matching technique introduced in Chapter 3, we formulate the tracking problem as a non-linear least squares problem, which is minimized with a gradient descent algorithm.

Parameterize the Movements

We assume the semi-static objects moves rigidly, thus their movements are represented by rotation matrices and translation vectors, denoted as $\{\langle R_i, T_i\rangle\}_{i=1}^{m}$. A vertex $\mathbf{v}$ on the semi-static model $\mathcal{S}_i$ is transformed as,

$$\tilde{\mathbf{v}} = R_i\mathbf{v} + T_i \tag{6.1}$$

The nonrigid movement on the dynamic surface $\mathcal{T}_j$ is modeled by embedded deformation

model introduced in Chapter 3 G_j. To deform a vertex $\mathbf{v}$ on $\mathcal{T}_j$, its k-nearest ED nodes on G_j is found, and a linearly blended affine transformation is applied,

$$\tilde{\mathbf{v}} := ED(\mathbf{v}; G_j); \tag{6.2}$$

where $ED(\cdot)$ is defined in Eq. 3.1.

Thus, the tracking problem is to estimate the rigid parameters $\mathcal{R} = \{\langle R_i, T_i \rangle\}_{i=1}^{M}$ and nonrigid parameters of $\mathcal{G} = \{G_j\}_{j=1}^{N}$. The parameters are estimated by solving the following non-linear least squares problem that has multiple energy terms,

$$\begin{aligned} \min_{\mathcal{R},\mathcal{G}} \; & w_{\text{reg}} \sum_j E_{\text{reg}}(G_j) + \sum_j E_{\text{rigid}}(G_j) \\ & + w_{\text{dns\_pts}} E_{\text{dns\_pts}}(\mathcal{R}, \mathcal{G}) + w_{\text{clr}} E_{\text{clr}}(\mathcal{R}, \mathcal{G}) + w_{\text{insct}} E_{\text{insct}}(\mathcal{R}, \mathcal{G}) \end{aligned} \tag{6.3}$$

where $E_{\text{reg}}(G_j)$ is the regularization term on the embedded deformation model and is defined in Eq. 3.5, and $E_{\text{rigid}}(G_j)$ is the global rigidity term defined in Eq. 3.6.

Similar as the data term defined in Chapter 3 for nonrigid alignment, $E_{\text{dns\_pts}}(\mathcal{R}, \mathcal{G})$ and $E_{\text{clr}}(\mathcal{R}, \mathcal{G})$ in Eq. 6.3 are terms that measure the geometry and color similarity of $\{\mathcal{S}_i\}$ and $\{\mathcal{T}_j\}$ with data observations at a frame under deformation of $\mathcal{R}, \mathcal{G}$. $E_{\text{insct}}(\mathcal{R}, \mathcal{G})$ is the energy term that prevents the self-interaction of the deformed dynamic models or its intersection with other objects. These three terms are introduced in the following two sections.

Geometry and Color Constraints

Obviously, surfaces $\{\tilde{\mathcal{S}}_i\}$ and $\{\tilde{\mathcal{T}}_j\}$ should align tightly with the point cloud observed by depth cameras at one frame under the optimal transformation. In addition, their colors should match with the observation as well. The dense point cloud alignment is measured by term $E_{\text{dns\_pts}}(\mathcal{R}, \mathcal{G})$, and the color similarity is measure by $E_{\text{clr}}(\mathcal{R}, \mathcal{G})$.

As in Chapter 3, We fuse all depth maps from Kinects at current frame into a volumetric representation, where each voxel records its distance $\mathcal{D}(\cdot)$ to the implicit surface. Thus, for a vertex $\tilde{\mathbf{v}}$ on the transformed models $\{\tilde{\mathcal{S}}_i\}$ or $\{\tilde{\mathcal{T}}_j\}$, $\mathcal{D}(\tilde{\mathbf{v}})$ measures its distance to the observed surface. Therefore, the overall alignment measurement is,

$$E_{\text{dns\_pts}}(\mathcal{R}, \mathcal{G}) = \sum_{v \in \{\mathcal{S}_i, \mathcal{T}_j\}} \left(\mathcal{D}(\tilde{\mathbf{v}})\right)^2 \tag{6.4}$$

where $\tilde{\mathbf{v}}$ has the form of either Eq. 6.1 or Eq. 6.2 depending on whether it is from a rigid surface $\mathcal{S}_i\}$ or a dynamic surface $\{\mathcal{T}_j\}$.

Similarly, $E_{\text{clr}}(\mathcal{R}, \mathcal{G})$ has the exact form as Eq. 3.11 for nonrigid alignmetn, *i.e.*,

$$E_{\text{clr}}(\mathcal{R}, \mathcal{G}) = \sum_{v \in \{\mathcal{T}_i, \mathcal{S}_j\}} \sum_k \left\| \delta_k\left(\tilde{\mathbf{v}}\right) \cdot \left[I_k\left(P_k\left(\tilde{\mathbf{v}}\right)\right) - c \right] \right\|_2^2, \tag{6.5}$$

where $\delta_k\left(\tilde{\mathbf{v}}\right)$ is the visibility term; $\delta_k\left(\tilde{\mathbf{v}}\right) = 1$ when $\tilde{\mathbf{v}}$ is visible to the k-th color camera, and 0 when invisible. Visibility checking is performed by projecting the surfaces $\{\mathcal{S}_i\}$ and $\{\mathcal{T}_j\}$ together to the image space and the vertices with smallest z values are visible. This visibility checking requires all the surfaces as inputs, which is another reason to employ the unified

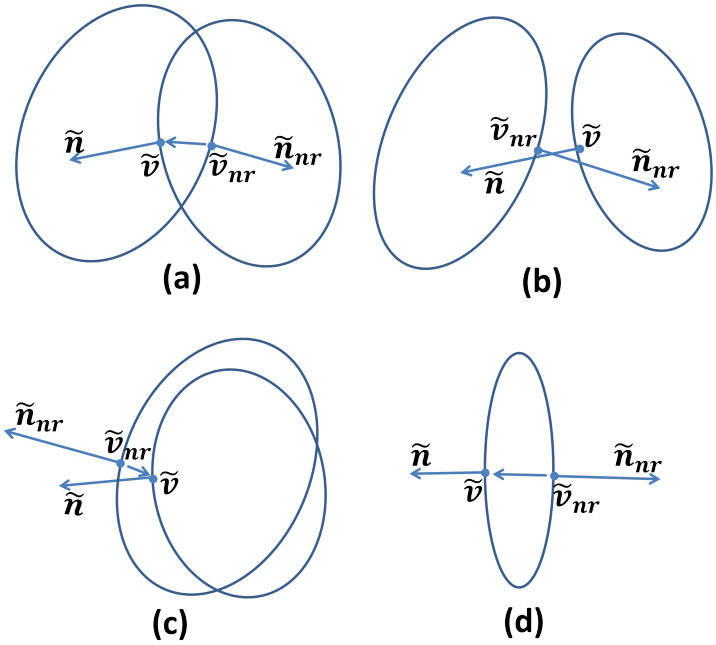


Figure 6.3: Various Intersection Situations. (a) positive intersection (intersection condition is satisfied); (b) non-intersection (intersection condition is not met); (c) deep intersection (intersection checking fails); (d) non-intersection (pass the intersection checking falsely).

tracking framework.

Intersection-Proof Constraint

Without further constraints, the surfaces being tracked tend to penetrate into each other during close interaction such as one sitting on a chair, as illustrated in Figure. 6.6. This is caused by severe occlusions during interaction and the high degree of freedom on the nonrigid movement of the dynamic objects. Self-intersection happens as well when different parts of the same dynamic object are close, *e.g.*, when one folding one's arms. We placed an intersection-proof constraint E_{insct} on dynamic objects in Eq. 6.3 to force intersecting surfaces to move

apart, *i.e.*,

$$E_{\text{insct}}(\mathcal{R},\mathcal{G}) = \sum_{v\in\{\mathcal{S}_j\}} \delta\left(\tilde{n}^T(\tilde{\mathbf{v}}-\tilde{\mathbf{v}}_{nr})>0 \;\&\; \tilde{n}_{nr}^T(\tilde{\mathbf{v}}-\tilde{\mathbf{v}}_{nr})<0\right)\left(\tilde{n}_{nr}^T(\tilde{\mathbf{v}}-\tilde{\mathbf{v}}_{nr})\right)^2. \tag{6.6}$$

$\delta(\cdot)$ is the intersection checking term. $\delta(\cdot)=1$ if the condition inside parenthesis is satisfied; $\delta(\cdot)=0$ otherwise. To detect whether a vertex $\tilde{\mathbf{v}}$ intersects others, we find its nearest point $\tilde{\mathbf{v}}_{nr}$ on other surfaces or parts. As shown in Fig. 6.3(a), vertice $\tilde{\mathbf{v}}$, $\tilde{\mathbf{v}}_{nr}$ and their outward normals $\tilde{n}$, $\tilde{n}_{nr}$ satisfy $\tilde{n}^T(\tilde{\mathbf{v}}-\tilde{\mathbf{v}}_{nr})>0$ and $\tilde{n}_{nr}^T(\tilde{\mathbf{v}}-\tilde{\mathbf{v}}_{nr})<0$ when intersection happens. Note that $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{v}}_{nr}$ are from the deformed surfaces $\{\tilde{\mathcal{T}}_i\}$ or $\{\tilde{\mathcal{S}}_j\}$ and deformed via Eq. 6.1 or Eq. 6.2. During optimization, E_{insct} pushes $\tilde{\mathbf{v}}$ moving toward the position where $\tilde{\mathbf{v}}-\tilde{\mathbf{v}}_{nr}$ is perpendicular to $\tilde{n}_{nr}$ ($\tilde{\mathbf{v}}$ is on the tangent plane at $\tilde{\mathbf{v}}_{nr}$), which is the direction departing the other surface.

In practice, we only search for the nearest point $\tilde{\mathbf{v}}_{nr}$ within some distance to $\tilde{\mathbf{v}}$ (15 cm in our experiment). For self-intersection prevention, we limit that $\tilde{\mathbf{v}}_{nr}$ must have a geodesic distance bigger than 10 cm and $\|\tilde{\mathbf{v}}-\tilde{\mathbf{v}}_{nr}\| << \|\mathbf{v}-\mathbf{v}_{nr}\|$ in order to prevent the situation shown in Fig. 6.3(d), where two vertices from the same surface part falsely pass the intersection detection.

There do exist positive intersection cases where the intersection checking conditions in Eq. 6.6 are not met, such as the deep intersection shown in Fig. 6.3(c). However, Eq. 6.6 is capable to detect and prevent intersections in most cases and easy to implement. Especially when enforcing Eq. 6.6 from the start, the deep intersections in Fig. 6.3(c) has little chance to happen.

We use the Levenberg-Marquardt algorithm to optimize Eq. 6.3. The Jacobians of the all

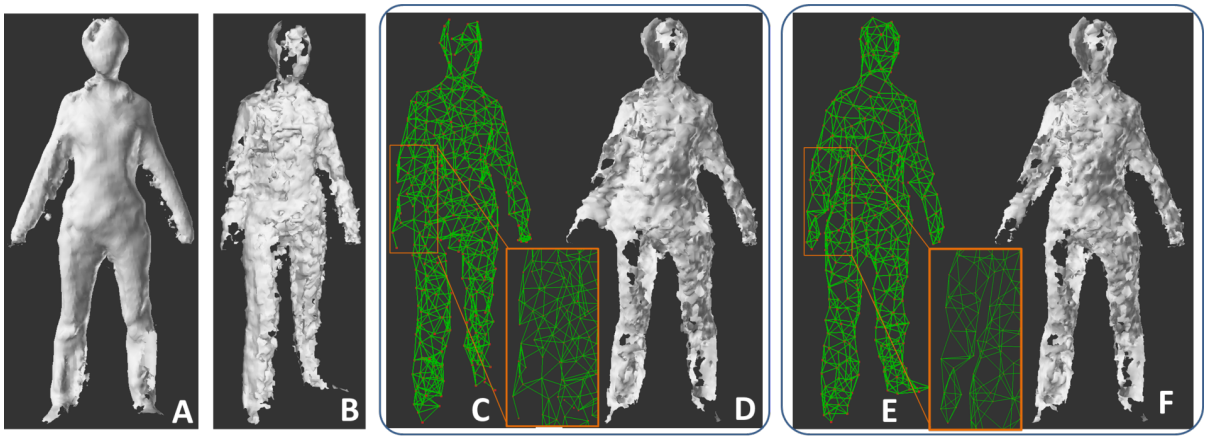


Figure 6.4: Backward Deformation Graph Generation. (A) previously accumulated surface at reference $\mathcal{S}^{ref}$; (B) currently observed surface $\mathcal{S}^{obv}$; (C) Deformation Graph on $\mathcal{S}^{obv}$ generated with random sampling; (D) deformed surface of $\mathcal{S}^{obv}$ (to be aligned with reference surface (A)) via the graph (C); (E) the deformed graph generated with our method; (F) deformed surface of $\mathcal{S}^{obv}$ via the graph (E).

the energy terms can be found analytically, and motion parameters of the previous frame are used for initialization.

6.2.3 System Pipeline

Since the static and semi-static objects are pre-scanned and their geometries are well constructed, we directly replace the live data by the prescan (for static background) or by the transformed prescan (for semi-static objects) based on the tracking result of Eq. 6.3. When a dynamic object enters the scene, the system segments it out and choose the frame with an open gesture (an example is shown in Fig. 6.4(A)) as its reference. This reference surface of the dynamic object along with other static and dynamic objects are aligned with following observations via the unified tracking framework. Next, the observations of the dynamic object are deformed back to the reference and accumulated at the reference. More specifically, the system performs the following steps at each new frame:

1. Fuse depth maps from 10 Kinects to one volume representation $\mathcal{D}^{obv}$; the surface representation $\mathcal{S}^{obv}$ is generated as well from $\mathcal{D}^{obv}$ via Marching Cubes algorithm.

2. If reference surfaces $\{\mathcal{T}_j^{ref}\}$ for dynamic objects exist, parameterize their deformation with embedded deformation model. Then run the unified tracking framework to align dynamic reference models (if any) and pre-scanned semi-static models with the observations ($\mathcal{D}^{obv}$ and $\{I_k(\cdot)\}$) at current frame via solving Eq. 6.3.

3. Perform scene segmentation on surface $\mathcal{S}^{obv}$. Each vertex on $\mathcal{S}^{obv}$ is assigned to one of a static model, a semi-static model, a dynamic model, or nothing. The reconstruction can be improved by replacing $\mathcal{S}^{obv}$ with static models, transformed semi-static models, and the deformed dynamic reference models. The vertex without an assignment is either noise or belongs to the new dynamic object. If a big connected component exists on the unlabeled surface, we treat this surface component as a new dynamic object and set the current shape as its reference.

4. For segmented dynamic surfaces $\{\mathcal{T}_j^{obv}\}$, compute the backward deformation ($\mathcal{T}_j^{obv} \rightarrow \mathcal{T}_j^{ref}$), and fuse the new observation to the references.

Note that we use a two-staged processing for dynamic surface accumulation: first a forward deformation from the reference surface $\mathcal{T}^{ref}$ to the current observation $\mathcal{T}^{obv}$ is estimated; then the backward deformation is calculated to deform the observation back to reference for data accumulation. This two-staged algorithm is the key to deal with topology variation by dynamic objects. The deformation graph sampled on surface $\mathcal{T}^{obv}$ under certain topologies might have an incorrect connectivity (*e.g.*, nodes on arms connect to nodes on the body trunk as shown

in Fig. 6.4(C)), leading to torn surface during deformation. Since we assume the reference frame has an "T"-pose, the sampled Deformation Graph at reference always has the correct connectivity.

Steps 1 and 2 in the above pipeline are straightforward; the other two steps are described below.

Scene Segmentation

Although the scene segmentation can be performed on the image (depth maps and color images) or even on the volume $\mathcal{D}^{obv}$, we choose to segment the surface $\mathcal{S}^{obv}$ because its by-product—vertex correspondence—can be used for backward deformation estimation of dynamic objects. For each vertex $\mathbf{v}^{obv}$ in $\mathcal{S}^{obv}$, we search for its corresponding vertex on the static background surface $\mathbb{B}$, or transformed semti-static or dynamic surfaces $\{\tilde{\mathcal{S}}_i\}$, $\{\tilde{\mathcal{T}}_j\}$ by considering distance, color similarity, and angle between normals . More specifically,

1. All the vertices on $\mathbb{B}$, $\{\tilde{\mathcal{S}}_i\}$, and $\{\tilde{\mathcal{T}}_j\}$ within distance δ ($\delta = 5$ cm in our experiment) from $\mathbf{v}^{obv}$ are listed as candidates of $\mathbf{v}^{obv}$'s correspondence. If the candidate list is empty, no correspondence is assigned for $\mathbf{v}^{obv}$ initially.

2. For each correspondence candidate of v^{obv}, we compute the two's distance d, color similarity p_{clr}, and angle between normals α. Among all candidates, we choose the one with largest sum of $\frac{d_{min}}{d} + p_{clr} + \cos(\alpha)$.

If a vertex on $\mathcal{S}^{obv}$ does not have a correspondence, but its neighbor does, we copy the neighbor's correspondence to that vertex.

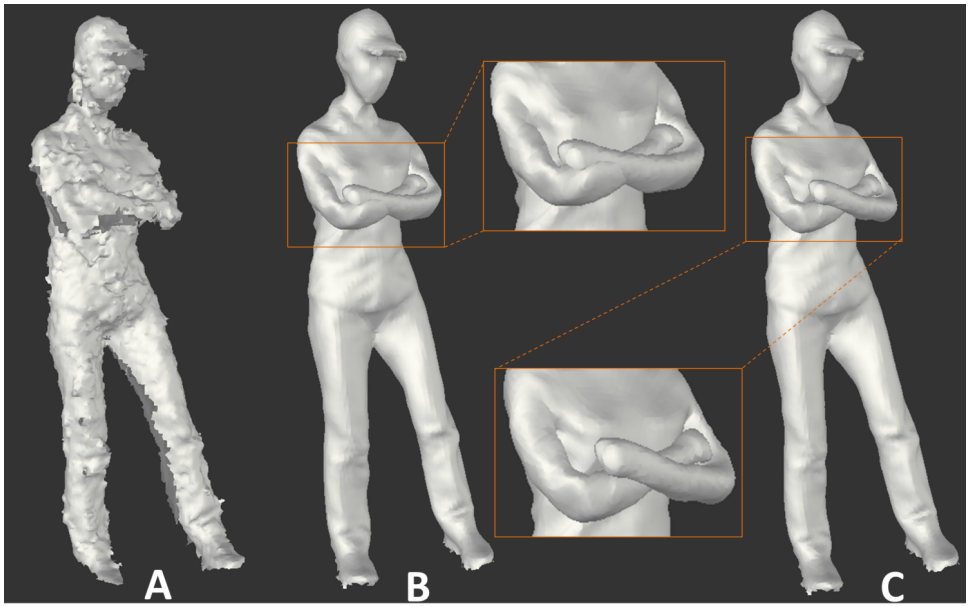


Figure 6.5: Tracking one person folding arms. (A) original input; (B) tracking without Intersection-Proof Constraint; (C) tracking with Intersection-Proof Constraint

Backward Deformation and Data Accumulation

The backward deformation is represented by an embedded deformation graph G^{bwd} on the observed dynamic surface $\mathcal{T}^{obv}$. As explained above, directly sampling a G^{bwd} from $\mathcal{T}^{obv}$ leads to problematic graph connectivities when the topology is inconsistent as shown in Fig. 6.4(C). Instead, we build a G^{bwd} with the same connectivity as the forward deformation graph $G^{fwd} = \{\langle g_i, A_i, t_i \rangle\}$ (the global rigid parameters are ignored for clarity). First, we transform G^{fwd}'s nodes position from $\{g_i\}$ to $\{g_i + t_i\}$. Note that $\{t_i\}$ is estimated during the tracking stage, and thus the new ED nodes are aligned with $\mathcal{T}^{obv}$. We then project $\{g_i + t_i\}$ to surface $\mathcal{T}^{obv}$ by finding their nearest surface points v_i^{obv}. We let $G^{bwd} = \{g_i = \mathbf{v}_i^{obv}\}$, and G^{bwd} has same nodes connectivity as G^{fwd}. Finally, we delete i-th node if $\|g_i + t_i - v_i^{obv}\| > \delta$. Fig. 6.4(E) shows projected G^{bwd} with correct graph connectivity.

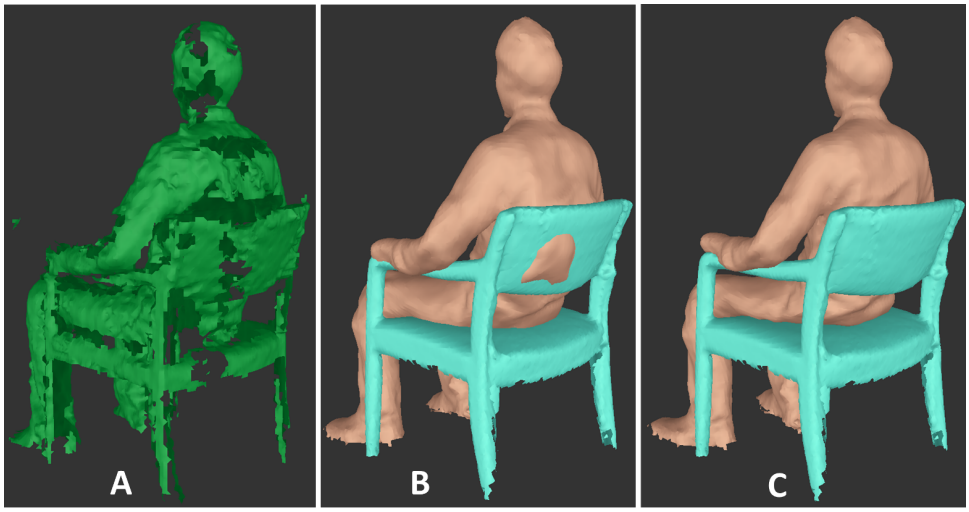


Figure 6.6: Tracking one person sitting on a chair. (A) original input; (B) tracking without Intersection-Proof Constraint; (C) tracking with Intersection-Proof Constraint

To deform a vertex v^{obv} on $\mathcal{T}^{obv}$ with backward deformation G^{fwd}, its k neighboring graph nodes on G^{fwd} need to be found. Due to the surface topology change, we cannot search them based on surface geodesic distance. Otherwise, a vertex of the arm might have a neighboring node on the body trunk. Because a vertex v^{obv} is associated with a vertex v^{ref} on the reference $\mathcal{T}^{ref}$ during scene segmentation, we let v^{obv} connected to the same graph nodes as v^{ref} (projected graph nodes on $\mathcal{T}^{obv}$) with the same weights. With the graph structure defined, the nonrigid deformation parameters of $\mathbb{G}^{bwd}$ is estimated using the method introduced in (Sumner et al., 2007) based on the estimated points correspondence between $\mathcal{T}^{obv}$ and $\mathcal{T}^{ref}$. Fig. 6.4(F) shows the result of deforming $\mathcal{T}^{obv}$ to the reference.

We used the same data accumulation algorithm introduced in Chapter 4, which is to transform the volume representation of $\mathcal{T}^{obv}$ and fuse it with the volume data at reference. One extension we made is that we build the color field together with the distance field. Since Kinect is essentially a RGB-D camera, each pixel have a color vector in addition to a depth value. When

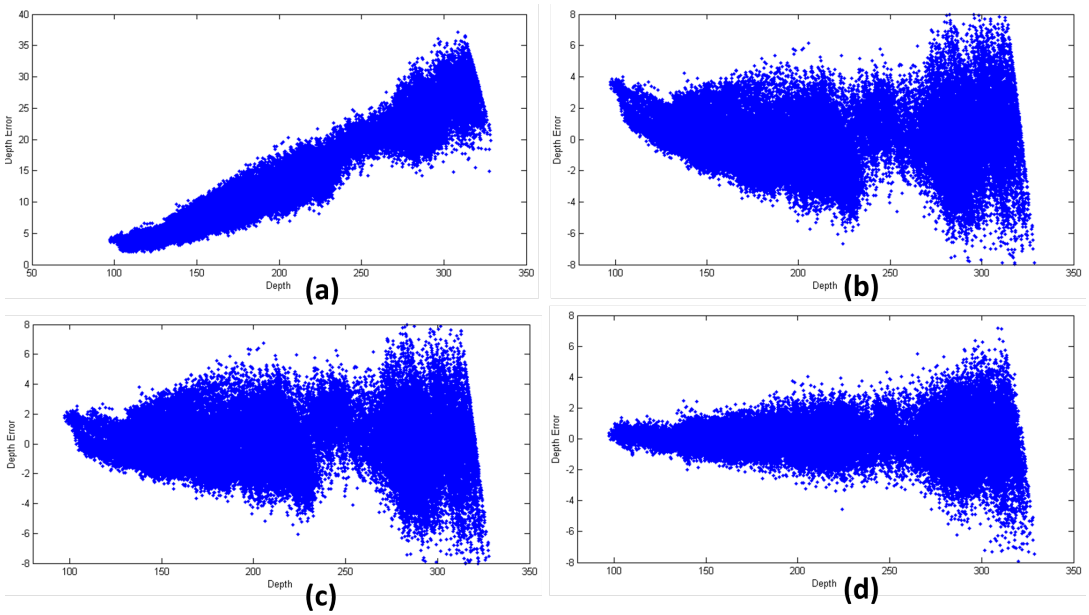


Figure 6.7: Depth bias v.s. depth. (a) the original depth bias without correction; (b) the depth bias after correction with a global linear function; (c) the depth bias after correction with a global quadratic function; (d) the depth bias after correction with our regional quadratic function.

building the distance field, we record a corresponding color vector along with distance value at each voxel. Color field observed at each frame is fused to the reference as well. A modified Marching Cubes algorithm interpolates the color for each vertex of the reference surface from the color field.

6.3 System Calibration

The system calibration is three-fold: 1) the calibration of Kinects' intrinsics and extrinsics; 2) Kinects' depth calibration; 3) alignment of prescan with the live data. We use the Microsoft Kinect SDK to align depth map to color images, so only the color cameras on Kinects require the calibration of intrinsics and extrinsics, which we achieved with the standard checkerboard calibration procedure.

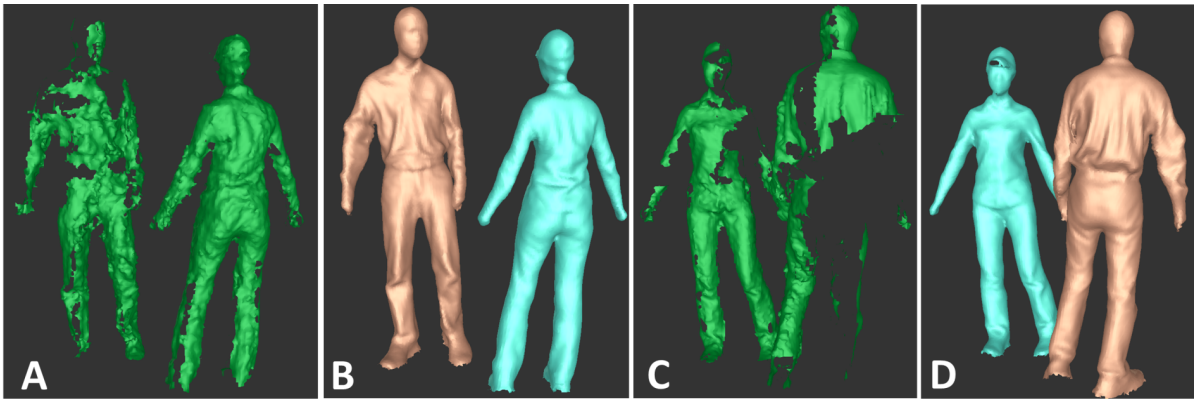


Figure 6.8: Tracking two persons where severe occlusion happens. (A)(C) original inputs; (B)(D) tracking results.

Also noticed by others, Kinect's output depth value deviates from the true depth, and the deviation is different at different areas on the depth map. Beck et.al. (Beck et al., 2013) built a 3D lookup table that covers the whole field of view of the Kinects to correct the depth bias; they use a motor to move a Kinect in the exact orthogonal direction to a flat surface and a commercial tracking system to track the distance of the Kinect from the surface. Instead, we designed a simpler calibration system using only the checkerboard while achieving comparable results.

Given the camera intrinsics and the detected 2D checkerboard corners, one can compute the 3D coordinates (x_c, y_c, z_c) in camera space precisely (Zhang, 2000). We denote the bias depth value obtained from the Kinect's depth map as z_d. We use z_c as the ground truth, and fit a quadratic function mapping z_d to z_c for various image regions: $z_c = az_d^2 + bz_d + c$. By waving a checker board in front of a Kinect at various positions, tons of $\langle z_d, z_c \rangle$ pairs are collected. We divide an 640×480 image equally into 20×15 blocks, and fit the quadratic function separately for each block using the corners fallen into each individual region. To guarantee the smoothness of parameters across the blobs, we let a block overlap with its neighbors by 16

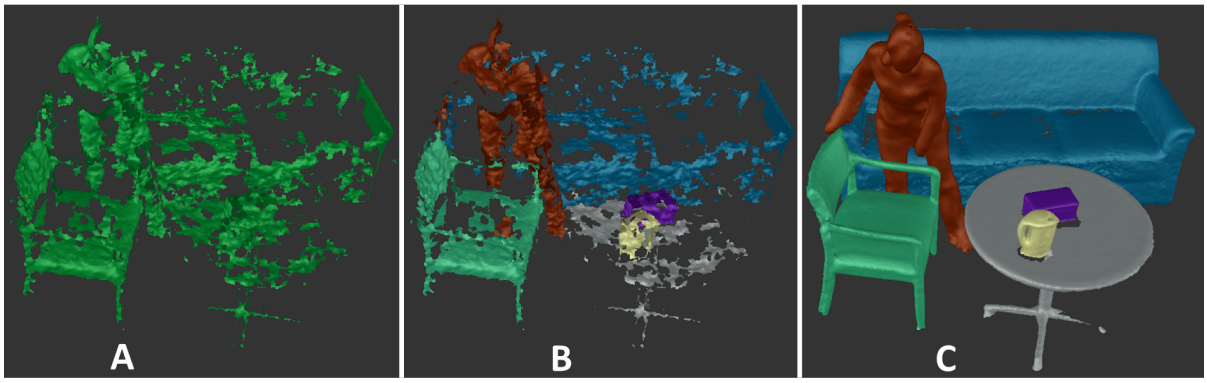


Figure 6.9: Tracking semi-static objects and dynamic objects. (A) original input; (B) scene segmentation result; (C) final models.

pixels during function fitting. Figure. 6.7 shows our depth calibration results along with the results of other methods.

To align the pre-scanned room with the live data from 10 Kinects, we capture the live RGB-D images of the empty room, and match these images with the key frames of the RGB-D sequence of the offline scanning. SIFT features and planes are used for matching. Afterwards, the refinement is performed using Eq. 6.3. In this case, instead of tracking semi-static and dynamic objects, we only track the pre-scanned room surface model.

6.4 Experiments

We mounted 10 Kinects in a room with the dimension of 4m$\times$5m$\times$3m (width, length and height), five of them covering upper space and the other five covering lower space. When mounting the Kinects, we try to minimize the field of view overlapping for adjacent units to minimize their interference. However, the interference still leads to large amount of missing data as shown in Fig. 6.9(A). The required calibration procedures are discussed in Section. 6.3, and the results are shown in Section. 6.4.1.

6.4.1 Results

The volume representation $\mathcal{D}^{obv}$ introduced in last section has the resolution of 1 cm (the size of each voxel is $1\times1\times1$ cm^3). It would occupy a huge memory for the size of our room. Instead, we eliminate the static background and only build $\mathcal{D}^{obv}$ for the remaining parts.

We tested our unified tracking frame work on multiple sequences under various situations. As shown in Fig. 6.5, our system solves the self-intersection problem decently. It also allow for close interaction between objects, such as one person sitting on a chair as shown in Fig. 6.6, since the Intersection-Proof term in Eq. 6.3 effectively pushes interacting surfaces apart.

Our system can also deal with the missing data problem due to occlusions and Kinects interference. In Fig. 6.8, dramatic data is missing for both people being tracked because of occlusion, but our system still tracks them robustly.

Fig. 6.9 shows an example of tracking both scanned semi-static objects and previously accumulated dynamic objects. The rendering results of the whole scene of the same frame is shown in Fig. 6.1(B)&(C). Our system successfully segments each objects from the noisy observation even though interaction exists between objects. The data accumulation on dynamic objects enhances their 3D quality dramatically as shown in Fig. 6.9(C) compared with Fig. 6.9(A).

6.5 Discussions

We designed a system for 3D reconstruction of room-sized dynamic environments. We divide the scene into three categories: static, semi-static and dynamic; and we use temporal

information in different ways for different categories. We also designed a unified tracking system that handles close interactions between objects. Finally, we show our system's capabilities on various situations.

Our system does have limitations. First, it is far from running in real-time, and it takes around 1 minute to process one frame under current CPU single-threaded implementation. We expect a future GPU implementation can improve the performance dramatically. Reducing the parameter space of the nonrigid matching is another possibility to boost the system performance and increase the system robustness. Second, even though the system allows the topology change of dynamic objects, it requires an open gesture for the reference frame. In our experiment, we ask people being captured to hold the open gesture until a reference frame is chosen.

CHAPTER 7: Conclusions

3D reconstruction of dynamic scenes can find many applications in areas such as virtual/augmented reality, 3D telepresence and 3D animation, while it is challenging to achieve a complete and high quality reconstruction due to the sensor noise and occlusions in the scene. This dissertation demonstrates our efforts toward building a 3D capture system for room-sized dynamic environments. A key observation is that reconstruction insufficiency (*e.g.*, incompleteness and noise) can be mitigated by accumulating data from multiple frames (*e.g.*, KinectFusion for static scene reconstruction). In dynamic environments, dropouts in 3D reconstruction generally do not consistently appear in the same locations. Thus, accumulation of the captured 3D data over time can fill in the missing fragments. Reconstruction noise is reduced as well.

The first piece of the system builds 3D models for room-scale static scenes with one handheld depth sensor, where we use plane features, in addition to image salient points, for robust pairwise matching and bundle adjustment over the whole data sequence.

In the second piece of the system, we designed a robust non-rigid matching algorithm that considers both dense point alignment and color similarity, so that the data sequence for a continuously deforming object captured by multiple depth sensors can be aligned together and fused into a high quality 3D model. We further extend this work for deformable object scanning with a single depth sensor. To deal with the drift problem, we designed a dense nonrigid bundle adjustment algorithm to simultaneously optimize for the final mesh and the

deformation parameters of every frame.

Finally, we combined the static scanning and nonrigid matching into a reconstruction system for room-sized dynamic environments, where we prescan the static parts of the scene and perform data accumulation for dynamic parts. Both rigid and nonrigid motions of objects are tracked in a unified framework, and close contacts between objects are also handled.

Note that we did not solve every aspect of the above reconstruction problem. Specifically, distinct limitations of the current work include lack of ability to deal with surface topology changes and system performance issues.

7.1 Algorithmic Contributions

Throughout the dissertation, we made various algorithmic contributions in the area of dense surface reconstruction.

7.1.1 Bundle Adjustment of Points and Planes

In Chapter 2, we demonstrate that plane features extracted from depth channel improve indoor environment reconstruction in various ways. Given that planes are the dominant feature for indoor environment, combining image feature and plane features leads to a robust pairwise matching across data frames. In addition, we designed a bundle adjustment algorithm that considers both point and plane features and optimize globally camera poses and point/planes in the world space.

7.1.2 Nonrigid Alignment Algorithm

In Chapter 3, we developed a robust nonrigid alignment algorithm. First, the algorithm works for general surface alignment in contrast to algorithms using kinematic models, as we parameterize the nonrigid motion with embedded deformation model and make no assumption on the surface. Second, the algorithm handles missing surface data decently, which makes it usable for partial-to-partial surface alignment. Third, we formulate the nonrigid alignment problem as an energy minimization problem with several least square terms, which enables it to be solved efficient by any least square solver. Finally, we incorporate both dense surface alignment measurement and color consistency to the formulation, and thus the alignment algorithm is robust for geometrically smooth surfaces.

7.1.3 Dense Nonrigid Bundle Adjustment

In Chapter 5, we demonstrated a bundle adjustment technique for dense nonrigid surface reconstruction. In contrast to classic bundle adjustment problem, where the correspondences are explicit and sparse, the structure is sparse, and the motion is rigid, our algorithm does not require *explicit* point-to-point correspondences, and it handles dense input data and optimizes jointly for both the dense mesh geometry and its nonrigid motion.

7.2 Developed systems in the disseration

During our pursuit for a room-sized dynamic scene reconstruction system, we had worked on various subproblems, which leads to several subsystems as byproducts that can work alone

to solve specific problems in the area of dense reconstruction.

7.2.1 Indoor Static Environment Scanning System

In Chapter 2, we designed a system for indoor environment reconstruction with one hand-held camera. ICP and SfM algorithms are expected to fail in challenging cases where a scene has very few image salient features and large geometrically smooth regions. In our system, we use both plane and image features for pairwise matching across frames and later for the bundle adjustment stage.

7.2.2 Dynamic Object Scanning and Tracking System

In Chapter 4, we developed a system for dynamic surface reconstruction with multiple depth sensors. The system does not require a prescan of the object being reconstructed. Instead, we build the model on-the-fly by accumulating data across frames. We do restrict the consistency of the surface topology during this data accumulation stage. Later, after a complete model is captured, we lift this requirement and only track surface deformation (*i.e.*, aligned model to new data observation).

7.2.3 Dynamic Object Scanning System with One Single Depth Sensor

In Chapter 5, we outlined a system that reconstructs a nonrigid surface with one single depth sensor. The system starts from a sequence of color images and depth maps. We first accumulate data over a short subsequence to extract partial scans. Next, we align partial scans consecutively to the reference pose; loop closures are detected; alignment error is distributed

over the whole loop. Finally, we run a dense nonrigid bundle adjustment step to optimize the desired dense geometry and nonrigid alignment parameters simultaneously.

Our system is template free (*i.e.*, no shape prior required), can be applied to fairly general surfaces (*e.g.*, articulated human bodies, clothes), allows for a significant amount of deformations (up to a fixed surface topology, see later limitations), and produces reconstructions of comparable quality to those of a static scanning algorithm (*e.g.*, KinectFusion).

APPENDIX A: DDF CALCULATION PSEUDO-CODE

Input: surface points $\{v_i\}_{i=1}^{M}$; DDF grid dimension $N_x \times N_y \times N_z$; DDF thickness μ.

```
1  d_xyz ← ∞; //signed distance field
2  Ind_xyz ← −1; //index to closest surface point
3  p⃗_xyz ← 0⃗ //directional field
4  //loop over M surface points
5  for(unsigned int i=0; i<M; i++) {
6    //loop over grid points within the bounding box (x1−x2, y1−y2, z1−z2)
7    //centered at v_i with radius of μ
8    for(unsigned int x=x1; x<=x2; x++)
9      for(unsigned int y=y1; y<=y2; y++)
10       for(unsigned int z=z1; z<=z2; z++)
11         d_new = distance to v_i;
12         if( d_new < d_xyz )
13           d_xyz = d_new;
14           Ind_xyz = i;
15 for(unsigned int x=0; x<N_x; x++)
16   for(unsigned int y=0; y<N_y; y++)
17     for(unsigned int z=0; z<N_z; z++)
18       if(Ind_xyz is surface boundary vertex)
19         continue;
20       //v⃗ is the closest surface point and n⃗ is its outward normal
21       //g⃗ is the current grid point
22       p⃗_xyz = v⃗−g⃗;
23       d_xyz = (p⃗_xyz · n⃗ < 0)?d_xyz : −d_xyz;
24       d_xyz = max(−1.0, min(1.0, d_xyz/μ) );
```

BIBLOGRAPHY

Agarwal, S. and Mierle, K. (2013). *Ceres Solver: Tutorial & Reference*. Google Inc.

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. (2005). Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 408–416.

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

Ballan, L. and Cortelazzo, G. M. (2008). Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT*, Atlanta, GA, USA.

Baran, I. and Popović, J. (2007). Automatic rigging and animation of 3d characters. In *ACM Transactions on Graphics (TOG)*.

Beck, S., Kunert, A., Kulik, A., and Froehlich, B. (2013). Immersive group-to-group telepresence. *Visualization and Computer Graphics, IEEE Transactions on*, 19(4):616–625.

Bojsen-Hansen, M., Li, H., and Wojtan, C. (2011). Tracking surfaces with evolving topology. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2012)*, 31(4).

Borrmann, D., Elseberg, J., Lingemann, K., and Nuhter, A. (2011). The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 02(3):1330–1334.

Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2006). Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proc. National Academy of Sciences (PNAS)*.

Butler, D. A., Izadi, S., Hilliges, O., Molyneaux, D., Hodges, S., and Kim, D. (2012). Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1933–1936. ACM.

Cashman, T. J. and Fitzgibbon, A. W. (2013). What shape are dolphins? building 3d morphable models from 2d images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):232–244.

Chen, J., Bautembach, D., and Izadi, S. (2013). Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(4):113.

Chen, J., Izadi, S., and Fitzgibbon, A. (2012). Kinêtre: animating the world with the human body. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*.

Crandall, D., Owens, A., Snavely, N., and Huttenlocher, D. P. (2011). Discrete-continuous optimization for large-scale structure from motion. In *CVPR*.

Cui, Y., Chang, W., Nöll, T., and Stricker, D. (2013). Kinectavatar: fully automatic body capture using a single kinect. In *Computer Vision-ACCV 2012 Workshops*, pages 133–147. Springer.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *SIGGRAPH*.

De Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S. (2008). Performance capture from sparse multi-view video. In *ACM Transactions on Graphics (TOG)*, volume 27, page 98. ACM.

de Aguiar, E., Theobalt, C., Stoll, C., and Seidel, H.-P. (2007). Marker-less deformable mesh tracking for human shape and motion capture. In *CVPR*.

Dou, M., Fuchs, H., and Frahm, J.-M. (2013). Scanning and tracking dynamic objects with commodity depth cameras. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 99–106. IEEE.

Dou, M., Shi, Y., Frahm, J., Fuchs, H., Mauchly, B., and Marathe, M. (2012). Room-sized informal telepresence system. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pages 15–18. IEEE.

Fitzgibbon, A. W. (2003). Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 31.

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). Manhattan-world stereo. In *CVPR*.

Gall, J., Stoll, C., De Aguiar, E., Theobalt, C., Rosenhahn, B., and Seidel, H.-P. (2009). Motion capture using joint skeleton tracking and surface estimation. In *CVPR*.

Gallup, D. and Frahm, J.-M. (2010). Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*.

Ghosh, A., Fyffe, G., Tunwattanapong, B., Busch, J., Yu, X., and Debevec, P. (2011). Multiview face capture using polarized spherical gradient illumination. *ACM Transactions on Graphics (TOG)*, 30(6):129.

Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., and Burgard, W. (2007). Efficient estimation of accurate maximum likelihood maps in 3d. *IROS*.

Gross, M., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Van Gool, L., and Lang, S. (2003). blue-c: a spatially immersive display and 3d video portal for telepresence. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 819–827. ACM.

Helten, T., Baak, A., Bharaj, G., Muller, M., Seidel, H.-P., and Theobalt, C. (2013). Personalization and evaluation of a real-time depth-based full body tracker. In *Proc. 3DV*, pages 279–286.

Henry, P., Krainin, M., amd X. Ren, E. H., and Fox, D. (2010). Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. *ISER*.

Herbst, E., Ren, X., and Fox, D. (2013). Rgb-d flow: Dense 3-d motion estimation using color and depth. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Hernández, C., Vogiatzis, G., Brostow, G. J., Stenger, B., and Cipolla, R. (2007). Non-rigid photometric stereo with colored lights. In *Proc. ICCV*, pages 1–8. IEEE.

Hirshberg, D., Loper, M., Rachlin, E., and Black, M. (2012). Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In Fitzgibbon, A., editor, *European Conf. on Computer Vision (ECCV)*, LNCS 7577, Part IV, pages 242–255. Springer-Verlag.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., and Davison, A. (2011). 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. ACM.

Kuster, C., Popa, T., Zach, C., Gotsman, C., and Gross, M. (2011). Freecam: A hybrid camera system for interactive free-viewpoint video. In *Proceedings of Vision, Modeling, and Visualization (VMV)*.

Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2011). Mav visual slam with plane constraintn. In *IEEE Int. Conf. on Robotics and Automation*.

Li, H., Adams, B., Guibas, L. J., and Pauly, M. (2009). Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia 2009)*, 28(5).

Li, H., Luo, L., Vlasic, D., Peers, P., Popović, J., Pauly, M., and Rusinkiewicz, S. (2012). Temporally coherent completion of dynamic shapes. *ACM Transactions on Graphics*, 31(1).

Li, H., Vouga, E., Gudym, A., Luo, L., Barron, J. T., and Gusev, G. (2013). 3d self-portraits. *ACM Trans. Graph.*, 32(6):187.

Liao, M., Zhang, Q., Wang, H., Yang, R., and Gong, M. (2009). Modeling deformable objects from a single depth camera. In *Proc. ICCV*, pages 167–174. IEEE.

Lieberknecht, S., Huber, A., Ilic, S., and Benhimane, S. (2011). Rgb-d camera-based parallel tracking and meshing. In *ISMAR*.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4).

Lourakis, M. I. (2010). Sparse non-linear least squares optimization for geometric vision. In *ECCV*.

Madsen, K., Nielsen, H., and Tingleff, O. (2004). Methods for non-linear least squares problems. *Lecture Note*.

Maimone, A. and Fuchs, H. (2011). Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 137–146. IEEE.

Maimone, A. and Fuchs, H. (2012). Reducing interference between multiple structured light depth sensors using motion. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pages 51–54. IEEE.

Mitra, N. J., Flöry, S., Ovsjanikov, M., Gelfand, N., Guibas, L. J., and Pottmann, H. (2007). Dynamic geometry registration. In *Proc. SGP*, pages 173–182.

Neumann, D., Lugauer, F., Bauer, S., Wasza, J., and Hornegger, J. (2011). Real-time rgb-d mapping on the gpu using the random ball cover data structure. In *IEEE ICCV/CDC4CV*.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE.

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

Pathak, K., Birk, A., Vaskevicius, N., and Poppinga, J. (2010). Fast registration based on noisy planes with unknown correspondesces for 3-d mapping. *IEEE Transactions on Robotics*.

Pathak, K., Vaskevicius, N., Poppinga, J., Pfingsthorn, M., Schwertfeger, S., and Birk, A. (2009). Fast 3d mapping by matching planes extracted from range sensor point-clouds. *IROS*.

Pokrass, J., Bronstein, A. M., Bronstein, M. M., Sprechmann, P., and Sapiro, G. (2013). Sparse modeling of intrinsic correspondences. In *Eurographics Computer Graphics Forum (EUROGRAPHICS)*.

Pollefeys, M., Gool, L. V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., and Koch, R. (2004). Visual modeling with a hand-held camera. *IJCV*.

Poppinga, J., Vaskevicius, N., Birk, A., and Pathak, K. (2008). Fast plane detection and polygonalization in noisy 3d range images. *IROS*.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42.

Schmidt, T., Newcombe, R., and Fox, D. (2014). Dart: Dense articulated real-time tracking. *Proceedings of Robotics: Science and Systems, Berkeley, USA*.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519–528. IEEE.

Sinha, S. N., Steedly, D., and Szeliski, R. (2009). Piecewise planar stereo for image-based rendering. In *ICCV*.

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring image collections in 3d. *ACM Transactions on Graphics*.

Snavely, N., Seitz, S. M., and Szeliski, R. (2007). Modeling the world from internet photo collections. *International Journal of Computer Vision*.

Sorkine, O. and Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Proc. SGP*, pages 109–116. Citeseer.

Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press.

Starck, J. and Hilton, A. (2007). Surface capture for performance-based animation. *Computer Graphics and Applications*, 27(3):21–31.

Steffen, R., Frahm, J.-M., and Forstner, W. (2010). Relative bundle adjustment based on trifocal constraints. In *ECCV*.

Sumner, R. W., Schmid, J., and Pauly, M. (2007). Embedded deformation for shape manipulation. *ACM TOG*, 26(3):80.

Taylor, J., Stebbing, R., Ramakrishna, V., Keskin, C., Shotton, J., Izadi, S., Hertzmann, A., and Fitzgibbon, A. (2014). User-specific hand modeling from monocular depth sequences. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Tevs, A., Berner, A., Wand, M., Ihrke, I., Bokeloh, M., Kerber, J., and Seidel, H.-P. (2012). Animation cartographyintrinsic reconstruction of shape and motion. *ACM TOG*, 31(2):12.

Tong, J., Zhou, J., Liu, L., Pan, Z., and Yan, H. (2012). Scanning 3D full human bodies using Kinects. *TVCG*, 18(4):643–650.

Valgaerts, L., Wu, C., Bruhn, A., Seidel, H.-P., and Theobalt, C. (2012). Lightweight binocular facial performance capture under uncontrolled lighting. *ACM TOG (Proc. SIGGRAPH Asia)*, 31(6):187.

Vlasic, D., Baran, I., Matusik, W., and Popović, J. (2008). Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics (TOG)*, volume 27, page 97. ACM.

Vlasic, D., Peers, P., Baran, I., Debevec, P., Popović, J., Rusinkiewicz, S., and Matusik, W. (2009). Dynamic shape capture using multi-view photometric stereo. *ACM Transactions on Graphics (TOG)*, 28(5):174.

Wand, M., Adams, B., Ovsjanikov, M., Berner, A., Bokeloh, M., Jenke, P., Guibas, L., Seidel, H.-P., and Schilling, A. (2009). Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data. *ACM TOG*, 28:15.

Weise, T., Bouaziz, S., Li, H., and Pauly, M. (2011). Realtime performance-based facial animation. *ACM TOG*, 30(4):77.

Weiss, A., Hirshberg, D. A., and Black, M. J. (2011). Home 3d body scans from noisy image and range data. In *Int. Conf. on Computer Vision (ICCV)*.

Wu, C., Stoll, C., Valgaerts, L., and Theobalt, C. (2013). On-set performance capture of multiple actors with a stereo camera. *ACM Transactions on Graphics (TOG)*, 32(6):161.

Ye, G., Liu, Y., Hasler, N., Ji, X., Dai, Q., and Theobalt, C. (2012). Performance capture of interacting characters with handheld kinects. In *Proc. ECCV*, pages 828–841. Springer.

Zeng, M., Zheng, J., Cheng, X., and Liu, X. (2013). Templateless quasi-rigid shape modeling with implicit loop-closure. In *Computer Vision and Pattern Recognition (CVPR)*, pages 145–152. IEEE.

Zhang, Q., Fu, B., Ye, M., and Yang, R. (2014). Quality dynamic human body modeling using a single low-cost depth camera. In *Computer Vision and Pattern Recognition (CVPR)*, pages 676–683. IEEE.

Zhang, Z. (2000). A flexible new technique for camera calibration. *TPAMI*, 22(11):1330–1334.

Zhou, Q.-Y., Miller, S., and Koltun, V. (2013). Elastic fragments for dense scene reconstruction. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 473–480. IEEE.

Zollhöfer, M., Nießner, M., Izadi, S., Rehmann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., and Stamminger, M. (2014). Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33(4).