A COMPUTER VISION BASED ULTRASOUND OPERATOR SKILL EVALUATION

by Zizui CHEN Thesis submitted

to the School of Graduate Studies in partial fulfilment of the

requirements for the degree of

M.Sc. Department of Computer Science

Memorial University of Newfoundland

November 2017

St. John's, Newfoundland and Labrador

Abstract

The aim of this thesis is to research inexpensive and automatic methods for analysing sonographers skill level, which reduces cost and improves objectivity. The current approach of teaching physicians to generate good quality ultrasound images is expensive and subjective, also takes significant time and resources, because it requires experienced instructors to guide and assess trainees in person. In this thesis, a distributed data collection system for synchronising and collecting data from multiple different sensors, including Microsoft Kinect 2 and ultrasound machine, was designed. Then hand movements are extracted from ultrasound images with an intensity-based image registration algorithm. The extracted movements data are analysed to find different patterns between novice and expert sonographers. A multi-sensor fusion algorithm is used in this thesis to extend the field of view of Microsoft Kinect 2, as well as overcome the cluttered environments and obstacles in clinics. Hand tracking is performed in the registered large point clouds with a semi-automatic colour-based segmentation algorithm.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

Many experts believe that ultrasound (US) is the stethoscope of the 21st century - a tool that extends the physical exam beyond the five senses [1]. It has been widely integrated into patient care with applications in many disciplines of medicine [2, 3, 4]. Evaluating sonographers skills to generate good quality ultrasound images takes significant time and resources [5]. At present, experienced sonographers observe trainees as they generate hundreds of images, provide them with feedback, and eventually decide if they have the appropriate skills and knowledge to perform the ultrasound test independently [6]. This current practice for evaluating a trainee is both expensive, due to the high salary of experienced sonographers, and subjective. The research in this thesis outlines the foundational work toward developing a tool that can provide a computerized evaluation of a sonographers skill level to reduce cost and enhance objectivity. This thesis focuses on analyzing the ultrasound for hand movements and tracking hands in 3D point clouds.

1.1 Objectives

The overall goal of the project is to analyse the hand movements of the trainee to determine the trainees skill level. This thesis aims to satisfy three objectives through the project:

O1 To develop a method of capturing and synchronising multiple channels of data, including ultrasound images and RGB-D images of the subjective. (Chapter 3)

O2 To evaluate the existing keypoint detectors and descriptors implementation in PCL and extending the effective field of view of a 3D computer vision system by using Microsoft

Kinect 2 sensors through an automated, calibration free, multi-sensor fusion algorithm that is used to capture the motion of the hands of the technician. (Chapter 4)

O3 To develop a semi-automated method of tracking an ultrasound operators hands in complex indoor environments in 3D space by using computer vision. (Chapter 5)

1.2 Thesis contribution

The main contributions of this thesis are:

1. The identification of key factors determining the ultrasound trainees skill level and a feature-free method for extracting hand movement from ultrasound images (Objective 1) (Published in [7]).

2. A calibration-free pair-wise point cloud registration algorithm, and the exhaustive evaluation of all keypoint detector and descriptor combinations in Point Cloud Library to test 3D reconstruction performance (Objective 2) (Published in [8]).

3. The development of a semi-automatic method for tracking hand movements in 3D space (Objective 3).

1.3 Thesis organization

Typically, the evaluation of an ultrasound operators skill levels can be done by either evaluating the transducer movement stability of ultrasound images or evaluating the patterns of hand movements. Chapter 2 presents a review of related works on clinical ultrasound evaluation and ultrasound image analysis techniques, followed by a review of literature related to automated hand movement techniques relevant to this work, including Kinect 2 performance evaluation, 3D point cloud registration and hand-tracking in 3D space.

Chapter 3 focuses on Objective 1 including 1. stability of transducer movement 2. sharpness of edges in ultrasound images 3. the position of big structures and 4. time to acquisition, with particular focus on the transducer movement stability analysis. A network-based distributed data collection system is also proposed in this chapter to overcome various performance limitations experienced during the data collection procedure. The contributions of this chapter were published at the *2015 International Conference on Image and Vision Computing New Zealand (IVCNZ)* in our paper titled *Feasibility of a semi-automated approach to grading point of care ultrasound image generation skills.*

Chapter 4 provides the work toward satisfying Objective 2. A review of the current techniques for extending the field of view of sensors, including both 2D-based and 3D-based methods, is presented, with a focus on 3D pair-wise registration, which is the most suitable approach for clinic environments. An exhaustive experiment is also described in this chapter that evaluates all available keypoint detector and descriptor pairs in PCL [9] to find out the best pair for this application. This work utilizes a benchmark public 3D dataset from Washington State University [10] to ensure the generalizability of the findings. The contributions of this chapter were published at *The International Symposium on Visual Computing, 2016* in our paper titled *Performance Evaluation of 3D Keypoints and Descriptors.*

Chapter 5 builds on the review of the current state of the art of appearance-based and model-based hand segmentation and tracking algorithms. Specifically, a novel semi-automatic method that tracks hand movement of a person in a 3D point cloud is presented. Chapter 6 concludes the thesis and proposes the potential future work to improve the current system.

Chapter 2 Literature review

2.1 Clinical ultrasound evaluation

Monitoring transducer movement as a measure of clinical skill has some applications in healthcare (e.g., surgical trainee knot-tying [11]). Specific to ultrasound skill development, Prinz et al. [12] have established that ultrasound technician skill attributes such as time to image acquisition and image quality improve with training. There have been many attempts to make ultrasound evaluation more objective. For example, Corretti et al. [13] defined a guideline for ultrasound assessment. Dubrowski et al. [11] proposed an assessment form to break down ultrasound tasks into discrete sub-components. Hammer et al. [14] proposed a scoring system for B-mode (BM) and power Doppler (PD) ultrasonography. Finally, Kimura [15] used a 7-point scoring system to classify parasternal long-axis (PLAX) ultrasound images as satisfactory or unsatisfactory. Unfortunately, all these evaluation schema depend on a human for scoring, which is subjective [11] and not directly suitable for automated implementation. Prinz et al. [11] have worked to formalize the evaluation process and make it more objective, but still requires a human for analysis, which is both subjective and costly.

2.2 Ultrasound image analysis

Ultrasound image quality is an important factor related to the assessment of operator skills [13]. Automated image quality analyses have been used in images captured using traditional photography. For example, [16, 17] use structural similarity to examine image quality and

determine photographers expertise, while [17] use peak signal-to-noise ratio (PSNR) as video quality metric. These methods rely on the clear structural patterns, or features, in images. However, ultrasound images are usually blurred and distorted [18]. Specic to ultrasound, several articles have been identied that discuss quality assessment and have been used to guide this research [19, 20, 21], but they are only suitable for analysing a single ultrasound image. The assessment of an ultrasound technicians ability requires a method of analysing the entire scan, which is comprised of a sequence of possibly thousands of consecutive images, preventing the direct application of existing methods.

The movement pattern of the transducer is another factor related to the assessment of operator skills. Operators movements have been routinely used for years for the purposes of generating 3D ultrasound images [22]. The movements can be extracted from a sequence of ultrasound images using image registration and speckle tracking techniques, again for 3D image generation [23, 24, 25]. However, this method has not been applied to technician skill assessment rather it requires the technician to move the probe within known and well-defined parameters.

Image stability has been deeply studied by many researchers with many published algorithms, particularly with respect to digital cameras and images [12, 26, 27, 28]. The common uses and most successful approaches are largely based on feature matching. Ultrasound imagery, however, is extremely noisy and dynamic, limiting the utility of feature matching techniques [18]. From the literature, an intensity-based image stabilization algorithm [29], which does not rely on features, represents the best opportunity for success since defined features are not present in ultrasound images.

2.3 Hand tracking using computer vision

Hand tracking in 3D space has been studied across a range of applications [30, 31, 32, 33, 34]. Donoser et al. [35] asserts that the methods for hand tracking can be categorized as either appearance-based hand tracking (e.g., [35, 36]) or model-based hand tracking (e.g., [34, 37]). The existing works mostly use 2D images as inputs, then segment and recover the hand position

in 3D space [30, 32, 37], with particular focus on identifying the positions of each finger [38, 39]. Other methods, such as the Kinect skeleton tracking, have also been used to track the hands position when most of the skeleton is visible [40, 41, 42], and more recently to provide an estimate of the hand pose [34, 43] and finger tracking [44].

However, in this project hand tracking is challenging because: 1. clinic environments are complex and dynamic; 2. the field-of-view of each camera is limited by obstacles (self- and environmental occlusions) and moving people; and 3. the visibility and resolution of the hands is poor because the camera is far from the hands. Skeleton tracking [40, 41, 42], which is by far the most robust 3D tracking approach currently available, as well as other 3D tracking approaches, are not suitable here because it cannot be guaranteed that the camera will see the entire body. Combining data from multiple sensors can help overcome the challenges faced in clinical ultrasound environments, particularly the first two. Approaches for combining (registering) data from multiple cameras are well defined for 2D images [45, 46], extending the overall field-of-view of the system and reducing occlusions. These approaches, however, generally require similar camera perspectives (i.e., marginal difference in the translation and rotation of the images), and 2D registration makes it difficult to recover the 3D coordinates in original space. Registering 3D images directly allows the sensors to be arbitrarily placed in the environment since the 3D data is perspective-independent. However, 3D registration algorithms are not developed beyond those that support incremental scene registration.

2.3.1 Multi-sensor fusion

Due to the complexity of clinic environments, the field-of-views of individual sensors are limited by moving people, random obstacles and cluttered environments. To extend the field-of-view of an overall system, multiple cameras need to be used. In a clinical environment, a calibration-free algorithm is required to fuse data from all cameras in 3D space. Calibration prior to system use is not feasible because clinicians are generally not trained to perform such tasks, or not willing to invest the time.

Data from a 3D sensor is typically represented as RGB-D data (normal RGB images with

an additional depth image which represents the distance between objects in the view and the camera), or point cloud (a set of data points in three-dimensional coordinate system)[47, 48]. Multi-sensor fusion is a process of registering multiple point clouds captured by a computer vision system to one global point cloud. The two main methods of 3D fusion are pair-wise registration [49] and Truncated Signed Distance Function (TSDF) [50]. TSDF is an incremental method for generating static 3D models in large scale [51], while pair-wise registration is suitable for registering multiple point clouds. TSDF is not suitable for this application because clinic environment is highly dynamic, while TSDF only works for static environments. Procedurally, pair-wise registration includes: 1. pre-processing both point clouds (filtering, down-sampling, etc.); 2. extracting keypoints from each point cloud; 3. computing feature descriptors for extracted keypoints; 4. finding correspondences between two sets of keypoints and descriptors; and 5. finding the transformation matrix which is most suitable for the correspondences [29]. The results are significantly affected by pre-processing methods and choosing parameters for keypoint extraction and descriptor computation. Finding correspondences and computing transformation matrices, however, have negligible impact on the performance of the algorithm ([8]).

Unlike 2D keypoint and descriptor identification algorithms, methods of identifying 3D keypoints and descriptors are not as robust and numerous and not as efficient or well-developed [52]. A widely adopted open source point cloud processing library, Point Cloud Library (PCL) [9], currently implements nine keypoint detectors, and 20 keypoint descriptors. Of these nine keypoint detectors, only five are suitable for unorganized point clouds (defined as point clouds where data are stored sparsely, like those provided by the Microsoft Kinect). These detectors are Harris3D [53], Harris6D [53], Intrinsic Shape Signatures (ISS) [54], Scale Invariant Feature Transform (SIFT) [55], and Smallest Univalue Segment Assimilating Nucleus (SUSAN) [56].

In our work [8], we discuss the PCL keypoint detectors and descriptors in detail. For completeness, I provide a summary here. The five main 3D keypoint detectors suitable for 3D unorganized point clouds in PCL are: Harris3D, Harris6D, ISS, SIFT and SUSAN. Harris3D [53] is derived from the traditional Harris detector [57] and uses surface normal for corner detection. Harris6D [53] extends Harris3D by combining both 3D and 2D information (intensity),

and removes weak keypoints using non-maximal suppression. ISS [54] is a highly discriminative local shape descriptor developed specifically for 3D point clouds. The 3D version of SIFT [55] extends the original 2D SIFT [58] by using 3D sub-histograms. Finally, the original 2D SUSAN [56] has been re-implemented as a 3D corner detector. The other three keypoint detectors, such as Normal Aligned Radial Feature (NARF) [59], Adaptive and Generic corner detection based on the Accelerated Segment Test (AGAST) [60] and Binary Robust Invariant Scalable Keypoints (BRISK) [61] [55] support range images or 2D point clouds only. Commonly used keypoint descriptors include: Persistent Histogram Features (PFH) [62, 63], which is a robust feature descriptor for 3D point clouds based on local geometry; Fast Persistent Histogram Features (FPFH) [62], which improves PHF by caching and reusing results in previous calculations and also reducing computation complexity by calculating the keypoint itself and its neighbours only; View Point Histogram [63] and Clustered View Point Histogram [64], which are expanded from FPFH to include viewpoint information; Rotation-Invariant Feature Transform [65], which is a descriptor extended from SIFT [58] and using colour information in the computation; Signature of Histograms of Orientations (SHOT) [66] and SHOTColour [67], which combine both signature and histogram for describing local features. Details on these and other descriptors can be found in the PCL literature [9].

Beyond the development of the detectors and descriptors themselves, researchers have begun to investigate their efficiency under certain real-world conditions. For example, Filipe et al. [55] conducted a comprehensive evaluation of the invariance of all 3D detectors available in PCL under various translations, rotations and scale changes by measuring their repeatability, which is the capacity of the detector to find the same set of keypoints in different instances of a particular model. They concluded that ISS was the most repeatable keypoint under various transformations. But keypoint detection is only a single step in the recovery and alignment process. Others have looked at the performance of both detectors and descriptors during the alignment process. For example, Alexandre [68] compared the object and category recognition of descriptors available in PCL with a single detector, the Harris3D [53] on a small subset of the RGB-D Object Database [10]. Hnsch et al [52] evaluated the multi-sensor registration performance of two detectors (NARF [10] [59] and SIFT [55]) with two descriptors

(PFH [69] and SHOT [66]) on a small set of 10 scenes. However, both studies are limited to testing a small number of detector/descriptor combinations, and only used small datasets for evaluation. Conversely, Moreels and Perona [70] evaluated the performance of several popular 3D detectors and descriptors available while finding matching correspondences in a moderate set of 100 objects viewed from 144 unique perspectives. While this study is more comprehensive in its inclusion of detectors and descriptors, it was conducted in 2007 before many mainstream implementations were available. Furthermore, the focus of the Moreels and Perona study was to investigate matching correspondences under an extremely wide range of rotations and translations, well beyond practical applications of 3D field of view extension and occlusion reduction.

Notably, the findings of the above studies suggest that the appropriate choice of a keypoint detector and a descriptor is generally sensitive to the application, and is impacted by changes in the scale, rotation and translation between different sensors [55]. The large number of possible combinations of 3D keypoints and descriptors suggest that the ideal pairing for any given application is difficult without knowledge of the performance of each pairing under the different transformation conditions (translations and rotations).

2.3.2 Three-dimensional hand tracking

As noted earlier, Donoser et al. [35] asserted that there are two main methods of hand tracking: appearance-based and model-based, and that the majority of the work in these areas utilizes 2D imaging. We extend the assertion of Donoser et al, given recent advances in 3D image processing to suggest that there are three common methods for tracking hands in three dimensions: skeleton tracking, appearance-based hand tracking and model-based hand tracking. Skeleton tracking is based on the reconstruction of the entire body skeleton, and hands are represented as single joints in the skeleton. Existing work on skeleton tracking includes: using surface estimation to recover the movement of the skeleton and possibly non-rigid temporal deformation of the 3D surface [71]; using whole body skeleton tracking and reconstruction like the Kinect SDK [72]; and using geodesic distances and optical flow to track human skeletons

[42]. These methods require most, if not all, of the body to be visible to the camera for skeleton reconstruction. This requires the ideal placement of the sensor(s), such that the tracker is given priority over other objects in the room (e.g., for a gaming system). In clinical settings, this is necessarily not the case. Rather, in clinical settings the clinician, patient and medical equipment are placed with priority, and any sensors must be placed inconspicuously to avoid interfering with any procedures. Accordingly, it cannot be assumed that most the body will be visible, or that the sensor perspective will be ideal. Instead, the hands must be tracked independently of the body in clinical applications.

Approaches that identify global hand locations in 3D space [73], [74] also need good visibility of the entire human body and require the depth sensor to be located at approximately above the ground, with the sensor viewing perspective oriented parallel to the ground. Recent work has extended this purpose to more unique viewing perspectives such as overhead [75] and egocentric [76] models.

Other approaches, including [77], [78], [79], track hand movements independently of the whole body, and focus on the hands in extreme detail. The main purposes of these approaches are identifying the movement of fingers, with the primary application of developing improved human-computer interaction. The cameras must be placed very close to the hands () [39] to capture the details of finger movement. Furthermore, these approaches are highly perspective- and orientation-dependent, and require that only the hands (and perhaps lower arms) are visible in the scene. This is currently an active area of research, but it is not directly relevant to our application since we cannot guarantee that the sensor will be located close to the hands, that the sensor orientation will be known, and that the operators hands will be dominant in the scene.

2.4 Summary

The current approaches for clinic ultrasound evaluation highly depend on human involvement, which leads to multiple issues such as expensive, slow and subjective training and evaluations.

Computerized technology can potentially assist the human in the evaluation process, increasing the objectivity and reducing the cost.

Computerized ultrasound evaluation methods can be approached from two perspectives: analysing the ultrasound image for image quality and analysing the operators hand movements for specific patterns. The operators hand movements can be inferred from transducer movements, which can be extracted from ultrasound video. However, this extraction process is tricky, because feature-based image registration methods do not apply here due to extremely noisy and dynamic images [18].

Clinic environments are cluttered, dynamic and complex. Thus, tracking operators hand movements is hard in clinic environments with traditional 2D based hand tracking methods, such as [34, 35, 36, 37]. A feasible solution to this issue is to track hands movements in 3D space, necessitating an extended system field of view using multiple sensors. This introduces a new issue: registering or combining data from these sensors.

Pair-wise registration is most suitable for dynamically registering multiple point clouds in real-time, which extends the field of view. The performance of registration is highly dependent on the choices of keypoint detectors and descriptors. Existing research on evaluating keypoint detectors and descriptors is limited in scope and exhaustiveness, or is outdated with the current state of the art of keypoint detection and description implementations. Accordingly, the best keypoint detector and descriptor pair for 3D registration is currently unknown.

Finally, existing hand tracking methods require clear images, and either need to place the sensor close to the hands (appearance- or model-based hand tracking), or require that most of the body parts are visible (skeleton tracking). The former methods are more focused on the micro-level, (e.g. the movement of each finger) instead of the hand position and orientation in 3D space. The later methods are designed for tracking hands as part of the whole body in 3D space, but require an ideal sensor placement, and most of the body parts must be visible to the sensor. Unfortunately, clinic environments are complex and highly dynamic. The sensors cannot be placed in an ideal position, and it cannot be guaranteed that the whole (or most of) the technicians body will be visible to the system.

The following chapters are focused on solving the issues highlighted above. Ultrasound trainees hand movements are extracted from ultrasound videos, and are tracked by 3D RGB-D cameras. To ensure the visibility of the hands, multiple 3D cameras are used to extend the field of view and avoiding obstacles.

Chapter 3 Ultrasound image analysis

This chapter outlines the design and implementation of a data collection system for capturing an ultrasound trainees scan data, as well as analysis of the ultrasound image to evaluate image quality.

Real-time ultrasound images were captured and recorded with a DVI to USB camera converter connected to a real ultrasound probe. The images were recorded as RGB images at [80]. The captured images from the ultrasound machine include both the ultrasound image of the target region and the user interface components of the ultrasound machine. Thus, it was essential to crop the captured images to remove the user interface components surrounding the image.

As discussed in sec.2.2, an ultrasound trainees skill level can be evaluated by using the stability of the transducer movement, which is more discriminative and easier to analyse using computer vision techniques.

The movements of the transducer can be classified into three categories: in-plane movement, out-of-plane movement and pivoting [81, 82]. In-plane movements are defined as the movements along the long-edge of the transducer. Out-of-plane movements are defined as the movements along the short-edge of the transducer. Pivoting is defined as moving the upper side of transducer while keep the transducer bottom in contact with the object. Out-of-plane movements and pivoting are harder to detect from sequential images than in-plane movement because the overlapped area between frames is smaller. When in-plane movement is the only movement, the distance of movement can be obtained by applying image registration techniques on the ultrasound images [83]. However, as discussed in sec.2.2, ultrasound images differ from photos captured by a regular RGB camera in several ways. Most notably, ultrasound images are blurry

making it hard to extract features, contain fewer distinct features and suffer from more noise [18]. Due to the above issues, feature-based image registration techniques are not efficient because in general only a small number of unreliable features are extracted. For images with fewer features like ultrasound images, intensity-based algorithms are more efficient than feature-based algorithms [29]. Accordingly, an intensity-based algorithm is used to first track the movement of objects in a series of ultrasound images, then to infer the movement of transducer. This chapter described how we fulfilled objective 1 (see chapter 1.1), to develop a method of automated ultrasound image capture and analysis facilitating operator performance evaluation. The network-based collection program developed for this work is attached as A. The analysis of recorded data is discussed in the following sections.

3.1 Data collection

A data collection system was designed to capture ultrasound images in a hospital room used for simulation and training. The equipment used in this experiment included: an ultrasound probe connected to a data acquisition system using a DVI to USB converter (Epiphan DVI2USB 3.0 [80]), two Kinect 2 depth sensors, installed on the left and right side of the testing area, and a third depth sensor installed on the ceiling above the testing area. An overview of the experimental setup is illustrated in fig.3.1.

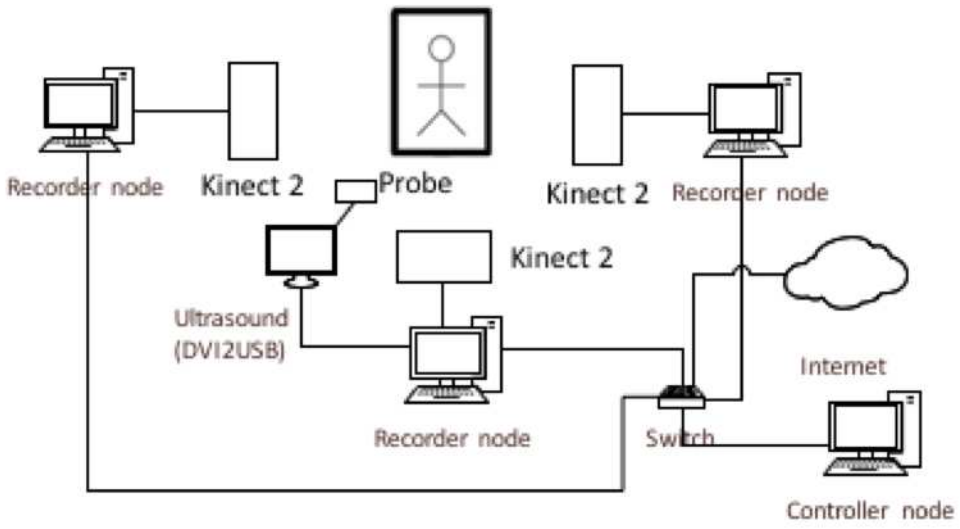


Figure 3.1: Experiement overview

Recording and synchronizing the raw data captured from the Kinect sensors presented many challenges including:

1. Disk I/O - The images returned from Kinect sensors are 1080p high-resolution images [84], requiring high disk bandwidth to write them to disk in real-time.
2. Processor usage - Images returned from Kinect sensors are encoded in the JPEG format. Decoding and converting these data to video is computationally intensive. This problem is compounded when processing images from multiple Kinect sensors concurrently.
3. USB bandwidth - Microsoft Kinect 2 sensors consume approximately halfof the bandwidth of a single USB 3.0 controller [85]. Furthermore, most desktop and laptop computers only have one USB controller.

Thus, a network-based distributed data collection system (fig.3.2) was designed to overcome these issues.

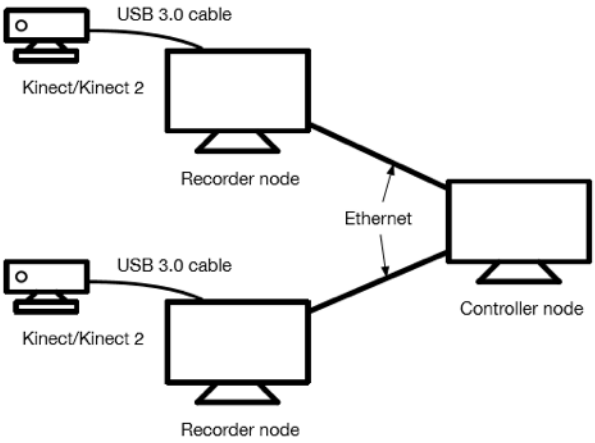


Figure 3.2: Network-based data collection system

The collection system consisted of two types of nodes: recorder nodes and a controller node. The recorder and controller nodes were connected through a local network via Ethernet cables. Recorder nodes connected to the Kinect sensors and ran the recorder program (See A), which listened to the commands sent from the controller node. Each recorder node could connect to one Kinect 2 sensor, or up to two Kinect 360 sensors (legacy support). Recorder nodes were running on a Macbook Pro (late 2013), which was equipped with an Intel Core i5 dual-core 2.4GHz CPU, CPU built-in Iris graphic card, 4GB memory and 128GB SSD. OpenCL

hardware acceleration was enabled for processing data from the Kinect 2 at the recorder node. The controller node ran a controller program (See A), which sent start and stop recording commands through the local network to all recorder nodes at the same time. The controller node was also a Macbook Pro (late 2013), with the same configuration as a recorder node.

Data retrieved from the sensors was processed and stored locally in the recorder nodes in the raw JPEG format. The controller node only sent out commands to start or stop recording. The controller node did not capture, store or process any data during data collection. Data synchronization was achieved by timestamping the frame in each recorder node. The system clocks of all recorder nodes were synchronized with Network Time Protocol (NTP) before recording began. This was accomplished by manually configuring each node to enable the built-in time synchronization mechanism [86].

3.2 Experiment

The experiment was performed with three experts and five novice operators (candidates). Each candidate ran a FAST (Focused Assessment with Sonography for Trauma) scan [87], which is a rapid bedside ultrasound examination performed by surgeons, emergency physicians and certain paramedics as a screening test for blood around the heart (pericardial effusion) or abdominal organs (hemoperitoneum) after trauma, on the test subject. All recordings were cropped to multiple videos clips, each one containing only one single step of the scan. The procedures of scans were recorded with the distributed data capture system proposed in the previous section, and the recordings were further converted to video files with FFMPEG [88], by interpolating the timestamped images. Each video file was manually decomposed into a series of discrete movements using the software StudioCode [89]. An advantage of using StudioCode was the ability to provide native support for synchronizing multiple videos with different frame rates, which was necessary to synchronize the data collected asynchronously from the three Kinect sensors.

This experiment focused on assessing the stability of transducer movement from ultrasound

video. As presented in sec.2.2, feature-based algorithms are not applicable to ultrasound images because the images are noisy and blurred. Preliminary experimentation suggested that only four or five features can be detected for most frames in an ultrasound video using the SIFT feature descriptor [58], which is less than the minimal number of keypoints required running FLANN [90]. Thus, an intensity-based image registration algorithm is proposed to calculate the movement between two frames. The workflow of the intensity registration algorithm is outlined in fig.3.3 [29].

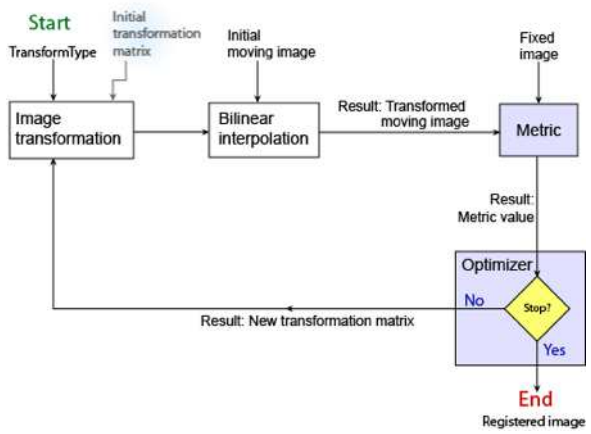


Figure 3.3: Intensity registration algorithm workflow

The algorithm is an iterative process, which requires a pair of images, an image similarity metric, an optimizer and transformation type to be specified. The image similarity metric defines the registration accuracy, allowing two images to be compared with a resulting scalar value that describes how similar the images are. The optimizer defines the methodology for minimizing or maximizing the similarity metric. The transformation type defines the type of 2D transformation that aligns the misaligned image (called the moving image) with the reference image (called the fixed image).

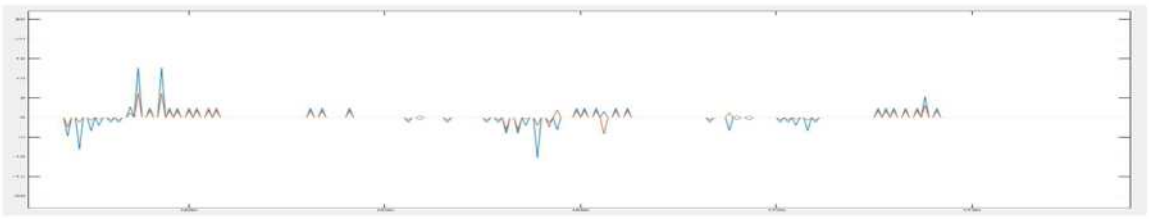
The process begins with a specified transform type (e.g. translation-only, rigid transform, similarity transform and affine transform) and an internally determined transformation matrix. The transformation is applied with bilinear interpolation to the moving image, determined by the transform type and the transformation matrix.

Then the similarity metric, mean squared error, is computed by comparing the transformed moving image to the fixed image. Finally, the optimizer is used to evaluate the stop condition,

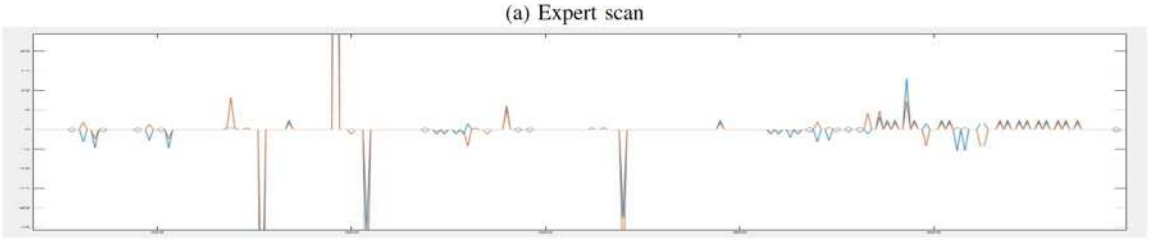
which ensures the algorithm terminates. The process stops when it reaches a point of diminishing returns or when it reaches the specified maximum number of iterations. Otherwise, the optimizer adjusts the transformation matrix to begin the next iteration. The optimizer used here is regular step gradient descent optimization [91]. It tunes transformation parameters to make the optimization follow the gradient of the image similarity metric in the direction of the extrema. The intensity-based algorithm computes in-plane movement between two frames.

3.3 Results

This section shows the analysis results from two clips, which consist one expert scan and one novice scan. Both scans are performed at the same region, on the same patient.



(a) Expert scan



(b) Novice scan

Figure 3.4: Transducer movement extracted by intensity-based image registration

The red lines and blue lines in fig.3.4 represent the ultrasound image movements along the X and Y direction, respectively. The expert scan includes 250 frames (from frame 1500 to 1750), and the novice scan includes 300 frames (from frame 400 to 700). It is common that expert operators scan faster than novice operators, because they are more familiar with the process and they can identify the target region and obtain good quality images faster than novice operators, explaining the temporal misalignment.

In the expert scan, the movements in the X axis remain stable to between 0-5 pixel/frame between frames 1540 1750, and there are negative movements in frames 1620 1650. In

the novice scan, there are 14 zero-crossings in the X direction and 14 zero-crossings in the Y direction between frames 400 and 470. Also, the movements vary between -25 to +25 pixels/frame. There are four large movements at frames 480, 500, 510 and 570 in the novice scan.

3.4 Discussion

In the expert scan fig.3.4, the stable movements in the x-axis indicates the transducer is moving along the x-axis at a steady speed. There are negative movements indicating the transducer moves backwards. The overall speed of the expert scan is 0-5 pixel/frame, which is stable. The five zero-crossings shows the direction of movement is consistent, with no significant back and forth movement.

In the novice scan, the 14 zero-crossings in both X and Y direction indicates the operator moves the transducer in an unstable way; the transducer is moving back and forth. The speed also changes much more significantly, ranging from -25 to +25 pixels/frame. Furthermore, the four large movements indicate that the image registration completely failed, because the novice operator lifted the transducer from test subject. Lifting transducer leads to a blank image, because an ultrasound signal cannot propagate into the body through air.

3.5 Conclusion

This chapter described the method for extracting the hand movements of the trainee from ultrasound videos and visually compared the stability of movement in novices and experts. The analysis of results focused on in-plane movements, which are the movements in X-Y plane. Although the results can be used for stability analysis and to further determine if the trainees skill level is novice or expert, the missing data for out-of-plane movements will still affect the accuracy.

The result may be further improved by adjusting the parameters of the intensity registration algorithm to achieve better registration results. The configuration of the ultrasound machine was also not necessarily optimized, and as such may be able to be improved to allow a consistent contrast and brightness during and across scans.

The current implementation still relies on a human to recognize the acquisition of the target region. However, it is possible to replace the human with an automated algorithm because the image quality is expected to adhere to the following pattern:

- before acquiring the target region, the image quality changes between low to high because the operator is trying to locate the region
- after acquiring the target region, image quality remains at a high level because the operator is viewing the target region from different angles, thus, the target region is kept in the image

This chapter of the thesis outlines the following contributions:

- an experiment for ultrasound probe movement data collection
- a general method to pre-process the collected data
- a method for tracking transducer in-plane movements using ultrasound images

Building upon the successes and limitations presented in this chapter, I now consider preliminary work toward tracking the transducer movements directly in 3D space with Kinect. Ultimately, these tracking data would be cross-referenced to the ultrasound image processing of this chapter.

Chapter 4 Multi-sensor fusion

Traditional computer vision-based object tracking technologies are not efficient in clinical applications because these environments are cluttered and dynamic, resulting in a limited field of view. A multi-cameras configuration overcomes some of the above issues; particularly those caused by occlusions of a single sensors field of view. However, camera calibration is required to create a unified scene from multiple sensors. Camera calibration is not practical in most clinic environments because technical personnel are not available, and clinicians are not trained for this task or are simply not willing to calibrate a system when entering each new room or following a room modification.

Accordingly, this chapter outlines the development of a 3D-based multi-sensor registration approach that does not require manual (human) calibration. The approach reconstructs a room with colour and depth images captured from multiple depth cameras. First, the colour and depth images received from Kinect 2 sensors are registered to individual point clouds. Then the individual point clouds are registered to a more complete global point cloud using incremental pair-wise registration [92, 93].

Pair-wise registration consists of the following steps [92]: 1. Extract keypoints from the original point clouds; 2. Compute keypoint descriptors for each keypoints in the original point clouds; 3. Find correspondences between the point cloud descriptors; and 4. Compute the 3D transformation matrix that fits the correspondences best. The performance of pair-wise registration is highly dependent on the first two steps: extracting keypoints and descriptors [8].

PCL implements nine keypoint detectors and 22 keypoint descriptors [94]. However, as shown in sec.2.3.1, only five detectors and 20 descriptors can be used with Kinect 2 point clouds.

The performance of the large number of detector/descriptor pairs available for 3D point clouds varies significantly based on the application. The performance of these detector/descriptor pairs has been explored in scenarios where the translation and rotation between multiple images or sensors is small (e.g. rotation less than 5 degrees) [55]. However, the performance of the detector/descriptor pairs is unknown in scenarios where the transformation between sensors is large. Accordingly, one important objective of this study is to extensively determine the performance of all detector/descriptor pairs available in PCL. From this evaluation, the most appropriate pair for our scenario, where the cameras can be placed arbitrarily in the room, can be determined.

4.1 Experimental setup

We designed an extensive evaluation experiment to exhaustively test the performance of each detector and descriptor pair available in PCL over a wide range of different translations and rotations. Candidate pairs were used to recover an artificial transformation on a large set of objects. The transformations were large enough to simulate most configurations in a common clinic deployment.

4.1.1 Dataset

The evaluation was performed on a large, publicly available RGB-D Object Database [22] from Washington University, which contains 300 household objects. The dataset is comprised of a video clip (and in some cases multiple videos) for each object, created by rotating the camera around the object from different angles. In this evaluation, the first frame of each video clip was selected. Thus, the dataset used in this evaluation included 300 RGB-D images of 300 household objects, respectively. Four sample objects from the dataset are shown in Fig 5. Note that the images are noticeably low in resolution, because the objects are small, (e.g. apples and bananas), and the camera was not placed very close to the objects. Rather, the objects

were cropped from a larger scene, and the resolution of the individual objects was restricted by the hardware limitation of the Kinect 360 [47] .

4.1.2 Methodology

We considered the cases of translation and rotation separately for each image. We translated each object from -100cm to +100cm in 5cm increments along the x, y and z axes independently. We then rotated each image from $-45°$ to $+45°$ in $15°$ increments around the x, y and z axes independently.

Figure 4.1: Sample objects from our dataset: ball; garlic; apple; coffee mug

The resulting transformation set was therefore 144 transformations for each detector/descriptor pair for each image. Using this transformation set, we manually transformed each source object per the source transformation matrix , creating a resulting target object. We then implemented each of the five detectors with each of the 20 descriptors (100 detector/descriptor pairs) on the source and target objects to attempt to recover the transformation matrix by aligning the target object to the source object. For each implementation, keypoints were extracted from both the source and target clouds using the detector, along with the associated descriptors. Correspondences were found using the Fast Library for Approximating Nearest Neighbours (FLANN) [25, 26, 27]. Random Sample Consensus (RANSAC) [28] was used for correspondence outlier removal and alignment of the source and target correspondences. All parameters were set to PCLs default, and we evaluated two sets of search radii for the keypoint detector and feature descriptors. We defined the small search radii as 3mm/5mm and large radii as 30mm/50mm for the detector/descriptor pairs. The error Err of the alignment was calculated per eq.4.1.

$$Err = \sum_{n} P_{s_n} - P_{t_n} \tag{4.1}$$

where P_s and P_t are point clouds before and after transformation, separately, and and are coordinates of the points in point clouds. The total evaluation set was then 144 transformations/pair $\times$ 100 pairs $\times$ 300 images = 4,320,000 samples.

4.2 Detector/Descriptor Pair Performance Evaluation

We defined a learned error threshold θ from experience for recovery of each source object from the associated target object on the 4,320,000 samples. Using eq.4.1, we defined a successful recovery as $Err < \theta$ and a failed recovery as one where $Err >= \theta$. To determine the effectiveness of a detector/descriptor pair over a given set of samples we defined an absolute and a relative success rate. The absolute success rate S_A was defined as the number of successful recoveries in the samples divided by number of samples, as in eq.4.2. The relative success rate S_R was defined as the number of successful recoveries in the samples divided by the difference between number of samples and number of failures, as in eq.4.3. The failure is further defined as the algorithm failed to compute a transformation matrix between original and transformed point clouds in some cases.

$$S_A = \#successful \quad recoveies/\#samples \tag{4.2}$$

$$S_A = \#successful \quad recoveies/(\#samples - \#failures) \tag{4.3}$$

For a given set of samples, the recovery alignment process could fail for the following reasons: 1. keypoint detection failure; 2. keypoint description failure; 3. correspondence estimation failure; and 4. too few correspondences for RANSAC alignment. For this reason, we defined the relative success rate S_R as the number of successful recoveries in the samples divided by total

number of recoveries for the samples. Due to the large number of detector/descriptor pairs, we only considered those with an absolute success rate higher than 0.5.

We further defined the invalid correspondence rate as the number of invalid correspondences over a given set of correspondences divided by the number of correspondences. We identified invalid correspondences by counting the number of rejected correspondences from RANSAC. We considered this invalid correspondence rate as well as the number of described keypoints and number of correspondences as measures of the absolute (S_A) and relative (S_R) performance of detector/descriptor pairs.

4.3 Runtime Environment

The substantial number of samples made serial or small-scale concurrent implementation of the testing prohibitive. Accordingly, we implemented the experiments on the ACENET Placentia computing cluster, a "3756 core heterogeneous cluster located at Memorial University"[29] as an array job.

4.4 Results

The data set was configured to run on the ACENET cluster as batches, with each batch containing all tests on 100 objects with processing executed concurrently in a queue utilizing approximately 40 cores at a time (determined dynamically by the ACENET scheduler) taking a total of 14 days to finish. The success rates S_R for the detector/descriptor pairs over all 4,320,000 samples with a learned threshold $theta = 10$ are shown in tab.4.1 for pairs with a success rate of over 0.5. The mean numbers of detected and described keypoints was equal in all samples, and are shown in tab.4.2a and tab.4.2b for all detector/descriptor pairs with a mean number of described keypoints greater than three over all samples for the small and large search radii. The mean numbers of successful correspondences for all detector/descriptor

pairs and invalid correspondence rates are shown in Table 2 and Table 3 over all samples for the small and large search radii for all pairs with an invalid rate less than 0.15.

Table 4.1: Relative success rate of detector/descriptor pairs with a success rate over 0.5

Keypoint	Descriptor	Small radius	Large radius
ISS	IntensitySpin	0.99	0.99
ISS	SHOTColor	0.94	0.94
ISS	SHOT	0.93	0.94
ISS	RIFT	0.88	0.71
ISS	ShapeContext	0.82	0.8
Susan	SHOTColor	0.76	0.77
Susan	SHOT	0.76	0.78
Susan	ShapeContext	0.59	0.62

Table 4.2: Mean number of described keypoints and invalid correspondence rate for the source and target objects for all detector/descriptor pairs with invalid correspondence rate less than 0.15

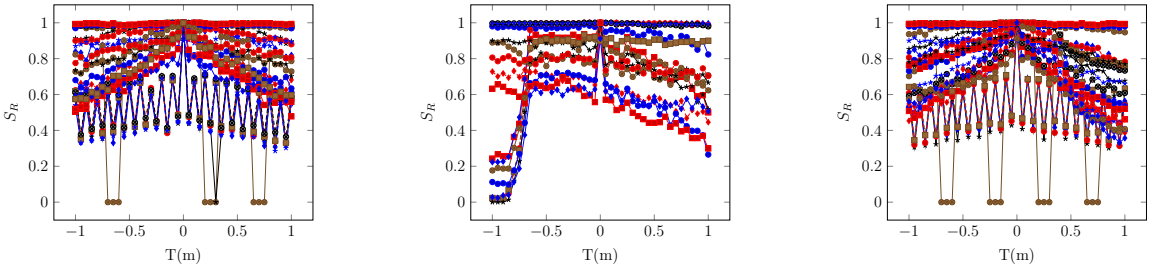
(a) small search radii

Keypoint	Descriptor	Keypoint(source)	Keypoint(target)	Invalid corrs
ISS	MomentInvariants	131.76	131.77	0
ISS	IntensitySpin	131.76	131.77	0.01
ISS	SHOT	131.76	131.77	0.01
ISS	SHOTColor	131.76	131.77	0.01
Harris3D	IntensityGradient	17.27	17.23	0.02
ISS	FPFH	131.76	131.77	0.03
ISS	IntensityGradient	131.76	131.77	0.03
Harris3D	FPFH	17.27	17.29	0.04
ISS	PFH	134.33	134.34	0.04
Sift	FPFH	5.6	5.64	0.04
Susan	SHOTColor	33.24	31.78	0.04
Sift	IntensityGradient	5.59	5.64	0.05
Sift	PFH	5.6	5.65	0.05
Sift	ShapeContext	5.6	5.64	0.05
Sift	SHOT	5.54	5.58	0.05
Sift	SHOTColor	5.54	5.58	0.05
Susan	FPFH	33.24	31.78	0.05
Sift	BOARD	5.54	5.58	0.06
Susan	SHOT	33.24	31.78	0.06
ISS	BOARD	131.76	131.77	0.07
ISS	ShapeContext	131.76	131.77	0.07
Sift	MomentInvariants	5.6	5.64	0.07
Susan	PFH	33.24	31.78	0.07
ISS	RIFT	131.76	131.77	0.08
Sift	PrincipalCurvatures	5.59	5.64	0.08
Susan	MomentInvariants	33.24	31.78	0.09
Susan	ShapeContext	33.24	31.78	0.09
Harris3D	BOARD	17.26	17.29	0.11
Susan	IntensityGradient	33.24	31.78	0.11
Susan	BOARD	33.24	31.78	0.14

(b) large search radii

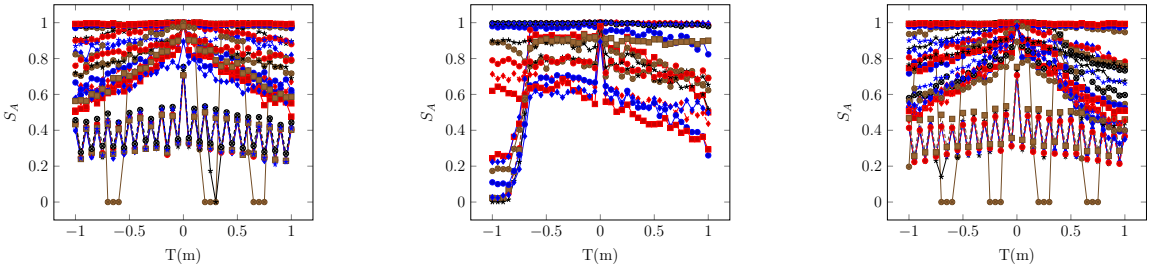
Keypoint	Descriptor	Keypoint(source)	Keypoint(target)	Invalid corr
Harris3D	Boundary	7.49	7.50	0.00
Harris3D	CVFH	17.17	17.21	0.00
Harris3D	PFH	5.67	5.77	0.00
Harris6D	Boundary	8.08	8.20	0.00
Harris6D	CVFH	18.40	18.55	0.00
Harris6D	PFH	6.38	6.44	0.00
ISS	Boundary	12.75	12.75	0.00
ISS	CVFH	143.45	143.45	0.00
ISS	SpinImage	131.76	131.77	0.00
Sift	BOARD	1.00	1.34	0.00
Sift	Boundary	1.00	1.34	0.00
Sift	CVFH	1.00	1.34	0.00
Sift	FPFH	1.00	1.34	0.00
Sift	IntensityGradient	1.00	1.34	0.00
Sift	IntensitySpin	1.00	1.34	0.00
Sift	MomentInvariants	1.00	1.34	0.00
Sift	PFH	1.00	1.34	0.00
Sift	PrincipalCurvatures	1.00	1.34	0.00
Sift	RIFT	1.00	1.34	0.00
Sift	SHOT	1.00	1.34	0.00
Sift	SHOTColor	1.00	1.34	0.00
Sift	ShapeContext	1.00	1.34	0.00
Sift	SpinImage	1.00	1.34	0.00
Susan	Boundary	8.77	8.51	0.00
Susan	CVFH	34.40	33.77	0.00
Susan	SpinImage	33.22	31.67	0.00
ISS	PFH	27.58	27.58	0.00
Susan	PFH	9.08	8.79	0.00
Harris3D	SpinImage	17.27	17.29	0.00
Harris6D	SpinImage	19.73	19.79	0.00
ISS	SHOTColor	138.50	138.49	0.00
ISS	SHOT	138.36	138.36	0.00
ISS	IntensitySpin	131.76	131.77	0.01
Harris3D	SHOT	17.09	17.13	0.02
Susan	SHOTColor	34.33	33.71	0.02
Harris6D	SHOT	18.36	18.52	0.02
Susan	SHOT	34.23	33.62	0.02
Harris3D	SHOTColor	17.18	17.23	0.02
Harris6D	ShapeContext	9.96	10.01	0.03
Harris6D	SHOTColor	18.42	18.58	0.03
Susan	ShapeContext	9.07	8.78	0.03
ISS	ShapeContext	27.58	27.58	0.03
Harris3D	ShapeContext	8.44	8.53	0.04
Harris6D	PrincipalCurvatures	18.46	18.61	0.05
Harris3D	PrincipalCurvatures	17.11	17.15	0.06
Susan	PrincipalCurvatures	34.24	33.61	0.11

We further consider the success rates for translations and rotations in (around) the x, y and z axes individually for the detector/descriptor pairs with sufficiently high mean success rates



(a) Success rate in x-axis (b) Success rate in y-axis (c) Success rate in z-axis

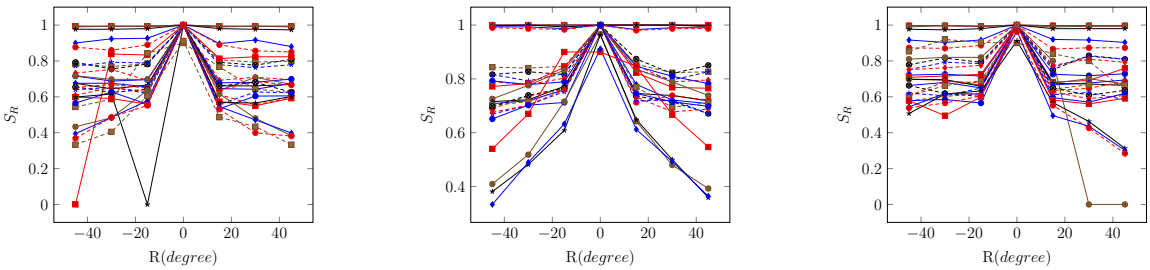
Figure 4.2: Absolute success rates for detectors/descriptors with mean success rates over 0.5 over the range of translations from -100cm to 100cm in the x-axis (a), y-axis (b) and z-axis (c).



(a) success rate in x-axis (b) success rate in y-axis (c) success rate in y-axis

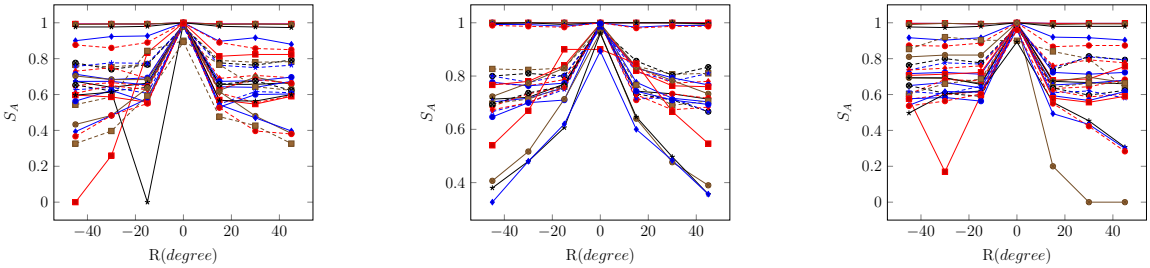
Figure 4.3: Absolute success rates for detectors/descriptors with mean success rates over 0.5 over the range of rotations from -45 to 45 in the x-axis (a), y-axis (b) and z-axis (c).

(tab.4.1). The absolute and relative success rates for translations are shown in fig.4.2 and fig.4.4 respectively. The absolute and relative success rates for rotations are shown in fig.4.3 and fig.4.5 respectively. The detailed versions of fig.4.2, fig.4.3, fig.4.4 and fig.4.5, with legends, can be found in the appendix.



(a) success rate in x-axis (b) success rate in y-axis (c) success rate in z-axis

Figure 4.4: Relative success rates for detectors/descriptors with mean success rates over 0.5 over the range of translations from -100cm to 100cm in the x-axis (a), y-axis (b) and z-axis (c).



(a) success rate in x-axis (b) success rate in y-axis (c) success rate in z-axis

Figure 4.5: Relative success rates for detectors/descriptors with mean success rates over 0.5 over the range of rotations from -45 to 45 in the x-axis (a), y-axis (b) and z-axis (c).

4.5 Discussion

The main contribution described in this chapter is the design and development of the calibration-free registration algorithm, and the comprehensive and exhaustive evaluation of all combinations of keypoint detector and descriptor available in the PCL for use with depth sensor data on an extensive 3D dataset. Individual detectors and descriptors have been evaluated in specific 3D applications (e.g.[52, 53, 55, 68, 70]) but the real performance of all possible detector/descriptor pairs has not yet been comparatively evaluated in the literature. The substantial amount of processing necessary to accomplish this evaluation was possible because of our access to the ACENET computational cluster which allowed significant use of concurrency across the data samples.

The results presented in tab.4.1 suggest that several detector/descriptor pairs have relatively high success rates over the entire dataset. These results are further supported by the data presented in tab.4.2a and tab.4.2b, with a direct correspondence between high success rate, large number of corresponding keypoints and low number of invalid correspondences. These data suggest that the success rate of detector/descriptor pairs is largely dependent on the number of keypoints detected. However, comparing the results for the small and large search radii (see tab.4.2a and tab.4.2b) supports that the number of keypoints is likely less important than the number of correspondences. For example, the pair ISS/MomentInvariants had an average of 132 keypoints for both small and large search radii, but a success rate during recover of 0.99 and < 0.5 for small and large search radii respectively. This is a result of the quality of

the keypoints which was poorer for the large search radii, preventing efficient correspondence estimation.

Inspection of the success rates in the axes individually (fig.4.2 and fig.4.3) under translation reveals that many detector/descriptor pairs are not translation invariant. Some pairs (e.g. ISS/MomentInvariant, ISS/SHOT) are translation invariant, achieving near perfect performance across all translations. Others (e.g. ISS/PFH, Susan/SHOT, Harris3D/IntensityGradient) have a performance that degrades linearly with increasing translation symmetrically around zero translation. Notably, all translations involving the SIFT 3D keypoint detector have a performance that is like the linearly degrading symmetrical performance with an additional cyclical modulation. The SIFT keypoint in PCL is the only implementation that utilizes voxel down-sampling, inherent to the original 2D SIFT algorithm. This down-sampling involves the computation of 3D voxels whose boundaries are impacted by floating point precision on translation. For example, a voxel with a boundary of zero, when translated by 70cm has a new, real boundary of 0.7m. A real point that is at 0m will fall into the voxel to the left of the down-sampling voxel under no translation. After translation, the boundary is represented in floating point as 0.69999, placing the point in the voxel to the right of the boundary. In this way, the down-sampling changes the point cloud, creating different keypoints and descriptors, ultimately affecting the correspondence and final recovery. The issue can be mitigated by implementing surface normal algorithm in double-precision. Accordingly, the current problems with the implementation of SIFT 3D must be addressed in PCL before its algorithmic performance can be evaluated. As evidenced in fig.4.2 and fig.4.3, some detector/descriptor (i.e. FPFH with ShapeContext/SHOT) pairs show asymmetrical performance around zero translation in the y-axis. This is the result of the fact that the PCL assumes surface normals always point toward the viewport origin. The objects, under negative translation, cause a virtual “flipping” of the surface normal for some surfaces, causing an undesirable behaviour in the detection and description of the objects keypoints. Setting the viewport to a very far location remediates this issue, transforming the results of the translations in the y-axis to mirror those of the x- and z-axis (i.e., translation invariance and linearly variant symmetry). Inspection of the performance of the detector/descriptor pairs shows similar performance in rotation compared to translation.

Some pairs (e.g. ISS/MomentInvariant, ISS/SHOT) are rotation invariant over the range. Others, (e.g., Harris6D/FPFH, Harris3D/FPFH) show a sharp degradation in performance over the first 15° of rotation symmetrically around zero rotation in all axes. However, this degradation plateaus between 15° and 45°. This phenomenon is a result of our dataset, which contains some objects that are symmetrical in nature. The performance degradation occurs for non-symmetrical objects almost immediately even under small rotations, but performance for these pairs does not change at all for symmetrical objects. The last set of pairs are rotation- variant, experiencing a continuous degradation in performance with increased rotation, symmetrical around zero rotation (e.g. Susan/IntensityGradient, ISS/ShapeContext).

Overall, considering the performance of all the detector/descriptor pairs over the varied objects, extensive test dataset, conditions and parameters, the ISS keypoint with SHOT, SHOTColor, FPFH, RIFT, MomentInvariants, IntensitySpin derivatives and SHOT descriptors performed the best. Under translation, ISS/IntensitySpin, ISS/MomentInvariants, ISS/RIFT, ISS/FPFH, ISS/IntensityGradient, ISS/SHOTColor and ISS/SHOT were considerably invariant, stable and constant over the entire range on all three axes. Furthermore, under rotation, ISS/MomentInvariants, ISS/IntensitySpin, ISS/RIFT, ISS/SHOT, ISS/SHOTColor and ISS/FPFH were invariant, stable and constant over the entire range around all three axes. From these data, it seems the most robust detector/descriptor pairs for 3D recovery or multi-sensor alignment are ISS/SHOT, ISS/SHOTColor and ISS/FPFH.

4.6 Conclusion

This chapter presents a comprehensive evaluation of the performance of various popular 3D keypoint detectors and descriptors currently available in the Point Cloud Library (PCL) to recover transformation information. The results show insight into which pairs work the best under various translations or rotations. After brute force testing all possible candidate pairs, we found the best pairs in both translation and rotation were ISS and SHOT or ISS and SHOTColor. However, the performance of both ISS/SHOT and ISS/SHOTColor pairs need to

be tested with real-world point clouds to make sure they can successfully reconstruct the clinic environment. Future work will look to evaluate these detector/descriptor pairs in this real-world context. Specifically, the real-world testing should be performed by placing two Kinect 2 side-by-side in a room and aligning the captured images together using the ISS/SHOT pairs. If successful, one Kinect 2 sensor should then be moved incrementally along one direction away from the second sensor which would remain stationary for the entire experiment. At each gradual increment, alignment should be attempted on the captured images from each sensor. This process evaluates the translation tolerance of the keypoint/descriptor pairs. The same process should then be applied to the remaining two axes of translation, and along the three axes of rotation independently. Finally, the study should be repeated with concurrent translations and rotations along multiple axes.

By using the keypoint/descriptor pair identified by the work presented in this chapter, it is theoretically possible to combine point clouds from multiple Kinect 2 sensors to extend the field of view of a single sensor, thus overcoming the limitations of a single-sensor system (e.g., occlusions). The resulting combined point cloud will be substantially more suitable for tracking the operators hand under the conditions present in cluttered and dynamic clinical environments.

Chapter 5 Hand tracking in 3D point cloud

The registered point cloud obtained from sec.4 is a large point cloud, which is comprised of many static and dynamic environmental objects, including humans and the background. Additionally, each object or human generally is a composition of several smaller objects or regions of interest. For example, as noted earlier, in a clinical setting, identifying and tracking the location of practitioners hands independently of the rest of the human body and amidst a cluttered and dynamic environment is highly meaningful. Accordingly, it is essential to segment the hands from the rest of the global point cloud to perform further analysis. The issue of hand segmentation is approached experimentally through the development of a novel and easy to implement semi-automatic 3D hand tracking algorithm.

5.1 Experimental Setup

The algorithm employs an iterative, semi-automatic process. Before using the system, one hand is selected manually from the global point cloud, initializing the tracker. This step is only completed one time. Next, the first frame of the 3D scene is segmented by both Euclidean distances and colour, identifying clusters/objects in the scene. In this way, different parts of any person in the scene are automatically segmented into clusters (e.g., body, head, arms and hands) along with other environmental objects. The cluster representing the manually segmented hand region is identified. Then, for each successive frame of new data, the point cloud is segmented into clusters automatically. The centre of mass of the hand in the previous frame is used to find the cluster that represents the hand in the current frame based on minimizing the 3D Euclidean distance over all potential clusters. This process is shown in fig.5.1.

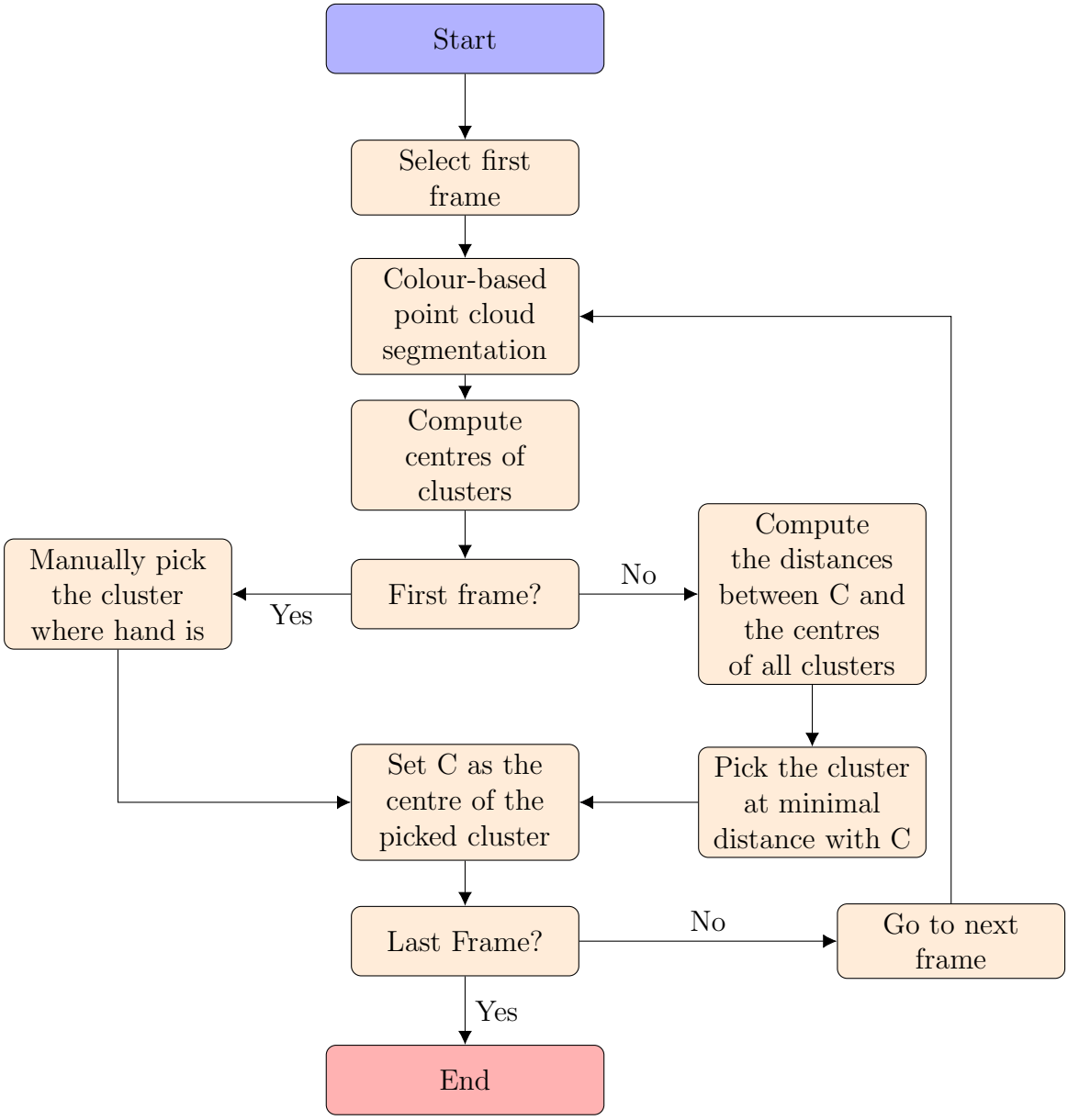


Figure 5.1: Workflow of hand tracking

5.1.1 Manual Hand Segmentation

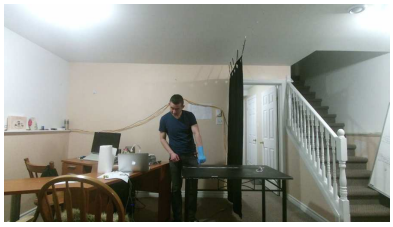
Two hand segmentation approaches were evaluated based on the available segmentation tools in PCL. The first was a 3D geometric region-growing approach [9]. In this method, a seed point is first selected from the global point cloud. The surface normal of the point is computed and compared to all nearest neighbours. If the angle between the seed and any neighbour is less than a defined threshold, the neighbour is added to the segmented object, the region is expanded, and all new neighbours are checked until no new neighbours are found. This process

is repeated for all points in the global point cloud until all points are part of a segmented object. The parameters for geometric region-growing were: minimum cluster size = 100; neighbours = 30; curvature threshold = 1.0; and smoothness threshold = 3.0 degrees. The second method utilizes the fact that hands are either skin colour or, in a clinical setting, wearing medical gloves (e.g., an ultrasound operator wears latex gloves during scanning). Accordingly, we employed colour-based region-growing [9, 95]. This approach is methodologically like geometric region-growing except for two differences. The first difference is that pixel colour is used instead of comparing the angle between the surface normal of a seed and its neighbour. The second difference is that after an initial segmentation, regions that have similar mean colour and are geometrically adjoined are merged to reduce over- and under-segmentation. After segmenting the hand, we use PCL built-in function to perform statistical outlier removal to remove the noise.

A static test was designed to evaluate the performance of the two different 3D segmentation approaches. In this test, a participant stood in the field of view of the sensors for a learned time of two seconds (30 frames), arms raised to the front and parallel to the ground, hands pointing up. After multiple trials ranging from 0.5 10 seconds, two seconds was selected as the trial time because longer times resulted in transient movements due to the difficulty of holding ones hands still for longer times. A colour-based region-growing algorithm built-in in PCL was used to segment the hand from other parts of the point cloud. The parameters of the algorithm were set as: distance threshold = 10 cm; point colour threshold = 5; region colour threshold = 3 for post segmentation merging; and minimum cluster size = 100 points.

Two Kinects were set up in a room at a height of 1m with 0.2m offset and 15 degrees rotation towards to each other. These positions maximized the view field of the test subjects. The sensor point clouds were automatically aligned using the method with ISS/SHOT for keypoints of multi-sensor fusion determined in chapter 4 to create a more comprehensive global reference frame. A participant in natural clothing and wearing blue gloves to simulate a clinician stood in the field of view of both sensors at a distance from the centre point of the sensors. The participant put one hand forward and a sample data frame was captured (fig.5.2a shows the colour image, fig.5.2b shows the depth image and fig.5.2c shows the generated point cloud).

The global point cloud was segmented using colour-based region-growing. In some frames, the hands cannot be segmented due to lighting condition and sensor noise. Those frames were not processed.



(a) RGB image

(b) depth image

(c) original point cloud

Figure 5.2: Colour-based hand segmentation

5.1.2 Automated Cluster Segmentation

The point clouds captured from the cameras include partial human bodies, other objects in the room, and the background scene. These complex point clouds can be decomposed into clusters, depending on their 3D positions and/or colour differences between points that comprise the clusters.

The PCL region-growing point cloud segmentation algorithm is extended from the original 2D version of algorithm [96]. The algorithm merges points that are similar in terms of the smoothness constraint. The output of this algorithm is a set of clusters, where each cluster is a set of points that are a part of the same smooth surface. Smoothness is computed by comparing the angles between the point normals.

The pseudo-code of this algorithm is listed in alg.1 [97]. All points are sorted by curvature values, from small to large, at first. The algorithm starts from the seed point which has the minimum curvature value (a flat area). The surface normal of each neighbouring point of the seed are then tested against the seeds normal. If the angle is less than a threshold value, then the point is marked as in the same cluster of the seed. Also, these points become new seeds, and repeat the previous algorithm, until seeds are empty, which means the region is fully grown.

Region-growing point cloud segmentation takes the continuity of points, or more exactly the continuity of the surface normal, as the only factor. A variant of region-growing segmentation is colour-based region-growing segmentation, which adds differences between the colours of neighbouring points into the algorithm [95].

```
input: Point cloud = P
       Point normals = N
       Points curvatures = c
       Neighbour finding function Ω(.)
       Curvature threshold C_th
       Angle threshold θ_th
1  begin
2      Region list R ← ∅;
3      Available points list A ← 1, ..., |P|;
4      while A is not empty do do
5          Current region R_C ← ∅;
6          Current seeds S_C ← ∅ Point with minimum curvature in A ← P_min;
7          S_C ← S_C ∪ P_min;
8          R_C ← R_C ∪ P_min;
9          A ← A P_min;
10         for i = 0 to size (S_C) do do
11             Find nearest neighbours of current seed point B_C ← Ω(S_C i);
12             for j = 0 to size (B_C) do do
13                 Current neighbour point P_j ← B_C j;
14                 if A contains P_i and cos^-1(|NS_C i, NS_C j|) < θ_th then then
15                     R_C ← R_C ∪ P_j;
16                     A ← A P_j;
17                     if cP_j < C_th then then
18                         S_C ← S_C ∪ P_j
19                     end
20                 end
21             end
22         end
23         Add current region to global segment list R ← R ∪ R_C
24     end
25     return R
26 end
```

Algorithm 1: Pseudo-code of region-growing algorithm

5.1.3 Seeding Automated Clustering

The hand tracking algorithm is designed to find a cluster in the current frame that is the closest, by Euclidian distance, to the cluster marked as the hand in the previous frame. This iterative process is suitable for tracking the movement of a specific cluster (e.g. hand cluster) in a sequence of point clouds. However, the algorithm cannot determine which cluster is the hand for tracking. Thus, an operator needs to manually pick the hand cluster on the first frame. In this application, the designed hand tracking system presents the point cloud of first captured frame, then prompts the user to visually identify and select the hand cluster as the start point.

5.1.4 Hand tracking in 3D point cloud

On each new frame F_i of video data after the first, we segmented all clusters from the global scene using both geometric and colour segmentation. The centroid and colour of the hand cluster for the previous frame F_{i-1} were then calculated, establishing the likely 3D location of the centre of the hand. Then, the centroids of all clusters in F_i were calculated, and clusters were ordered by their distance from the centroid of the hand cluster in F_{i-1}. The colours of these clusters were compared to the colour of the hand identified in F_{i-1}, starting with the nearest cluster. The first matched cluster was identified as the new centre of the hand. The pseudo-code for the 3D hand tracking is summarized in alg.2.

5.1.5 Hand motion modelling and characterization

Segmenting and locating the hand in 3D space is useful for detecting interactions with other humans and environmental objects. However, from a clinical perspective we are also interested in characterizing practitioners hand motions. We consider the clinical task of tracking the hand motion of a sonographer trainee to automatically and objectively assess his or her level of skill while holding the ultrasound transducer and performing an ultrasound scan. Preliminary data provided by clinicians skilled with ultrasound technique defined the preliminary motion

```
1  begin
2      cluster = show(segment(frames[0]);
3      centroid = getCentroid(cluster);
4      colour = getColour(cluster);
5      for I do=1 to length(frames) do
6          clusters = segment(frames[i]);
7          centroids = new List;
8          for cluster in clusters do
9              centroids.append(getCentroid(cluster));
10         end
11         Order by distance(centroids, centroid);
12         for j do=1 to length(centroids) do
13             currentCluster = getClusterByCentroid(centroids[i], clusters);
14             if colourMatch(currentCluster, colour) then
15                 centroid = getCentroid(hand);
16                 colour = getColour(currentCluster);
17                 break;
18             end
19         end
20         Write(hand);
21     end
22 end
```

Algorithm 2: Hand tracking algorithm

parameters of interest to our study as: time to target, velocity and stability of movement. However, due to the exploratory nature of this study we focus on velocity and stability of movement. As determined in sec.5.1.4, the centroid of the hand cluster represented the hands 3D location. Each frame of data was time stamped, allowing the centroids movement to be interpreted as a velocity over time. However, due to the hardware variations of Kinect devices, the frame rate of each sensor was slightly variable and different between sensors. In this thesis, the global frame rate was set at 30 FPS, and the centroids in the global reference frames were interpolated to their nearest global time.

The velocities of the hands were calculated in each of the Cartesian axes (X, Y, and Z direction) every 30 frames (1 second) using the interpolated 3D positions and times. The overall velocity was also calculated over the 3D Euclidean distance for each interpolated time. To smooth the result, velocity was averaged for every 5 frames (1/6 second). Stability of movement was inferred from the acceleration, or rate-of-change of the velocity. Stable movement (smooth acceleration) was defined as accelerations $< 5cm^2/frame$.

The experimental design was as follows:

1. Participants wore blue medical gloves and stood in front of a desk in a cluttered room.
2. A tape measure was placed on the desk to provide a ground truth distance measure.
3. Participants were asked to:
   (a) Raise one hand above the tape measure to a starting position;
   (b) Move his/her hand to the right, along X-direction to a target distance which set by the specific experiment;
   (c) Moves his/her hand to the left, along X-direction and back to the start position.
4. Step 3 was repeated in the Y-direction (upwards).

5.2 Results and discussion

The experiments included two participants, each of them performing a series of movements in front of a Microsoft Kinect 2. The movements were measured directly through image data captured by the Microsoft Kinect 2, and compared to the measurements from the tape measure.

5.2.1 Automated Cluster Segmentation

The subject was segmented into multiple clusters, including face, left and right hands. Segmentation was based on colour-based region-growing segmentation. To enhance the result, the subject was wearing a blue medical glove.

5.2.2 Seeding and Automated Clustering

The hand cluster was manually selected from the segmented point cloud as in fig.5.3. The original point cloud is shown in fig.5.2c, and segmented point cloud is shown in fig.5.3 (left), each colour represents a cluster. The selected cluster (hand) is shown in fig.5.3 (right).

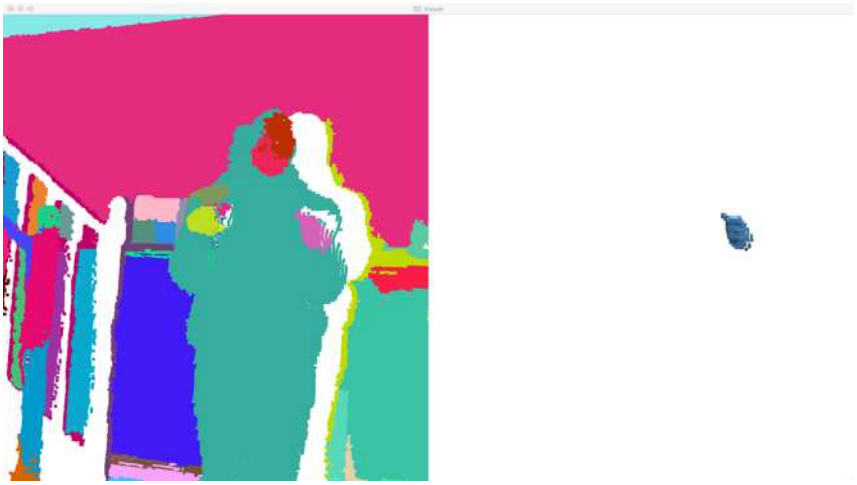


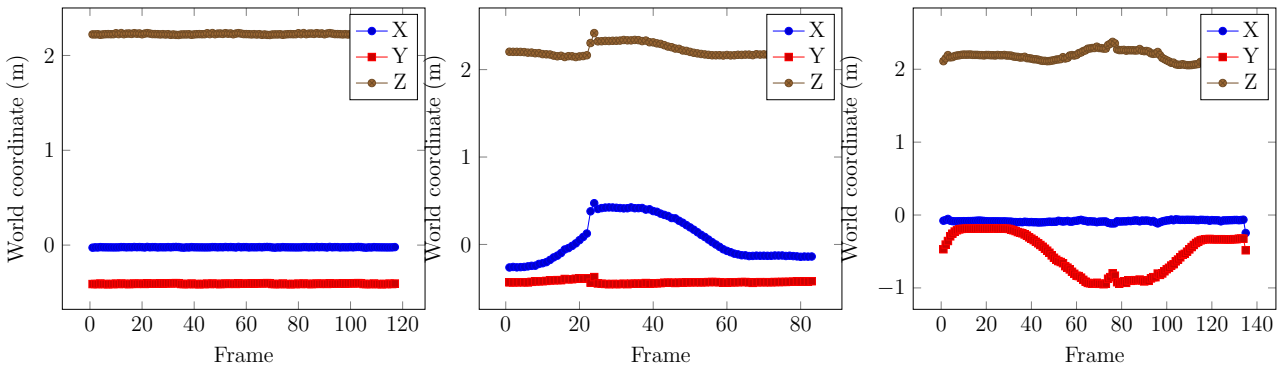
Figure 5.3: Colour-based hand segmentation with selected hand cluster (left: segmented point cloud, right: selected hand cluster)

5.2.3 Hand tracking in 3D point cloud

Subject 1

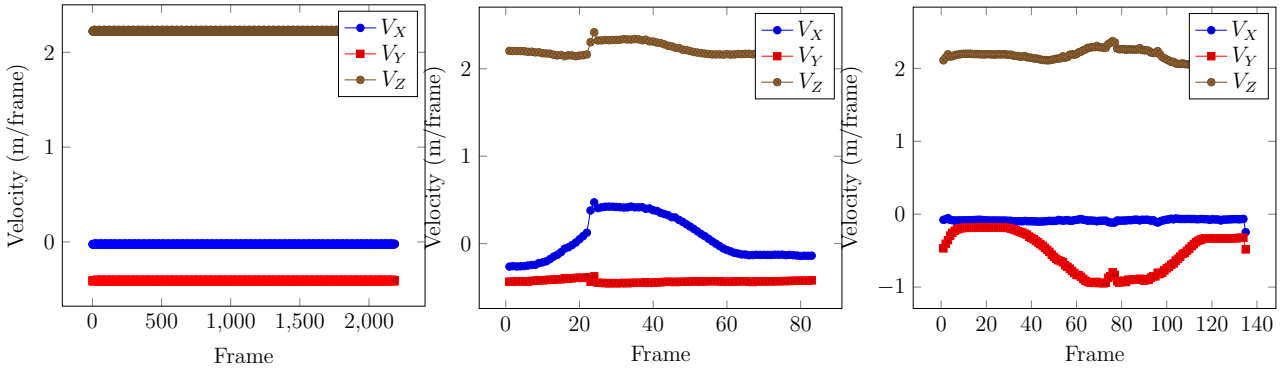
The colour segmentation stability results can be seen in fig.5.4a and fig.5.5a. A sample frame of this test is shown in fig.5.8. The participant holds the hand visually still, the results are stable to within variance of $2.86 \times 10^{-5} m^2$, suggesting that the centre of mass of the hand point cloud over time corresponds to the actual movement of the hand. To evaluate the effectiveness of the colour-based region-growing segmentation the participant again put one hand forward to a start point at $(-0.13m, -0.42m)$, then moved the hand from start point to right at $(0.42m, -0.45m)$, and back to start point (x-test). The results of x-test are shown in fig.5.4b and fig.5.5b. The participant then put the hand at $(-0.07m, -0.33m)$, lowered the hand down to $(-0.07m, -0.94m)$, and returned to start point (y-test). The results of y-test are shown in fig.5.4c and fig.5.5c. The depth (z) coordinate was not considered for this test because the x and y coordinates are inferred from z during world coordinate recovery, thus there is no need for a separate z test.

For example, when evaluating motion in the x-axis, the x-component of the hand motion increased from -13cm, reached a minimal at frame 57 (0.42m), then steadily decreased to -0.26m. In test y, the y-component reached a maximum at frame 67, maximum at -0.94m, then



(a) Stability test (b) x-test (c) y-test

Figure 5.4: 3D coordinates of the centre of mass of the hand cluster in world coordinates (experiment 1)



(a) Stability test (b) x-test (c) y-test

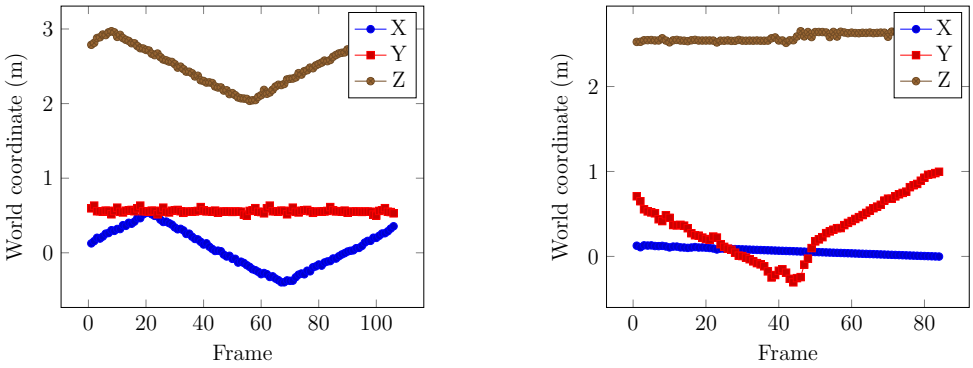
Figure 5.5: Velocity of the centre of mass of the hand cluster in world coordinates (experiment 1)

decreased to -0.4cm, supporting 3D colour-based region-growing segmentation.

Subject 2

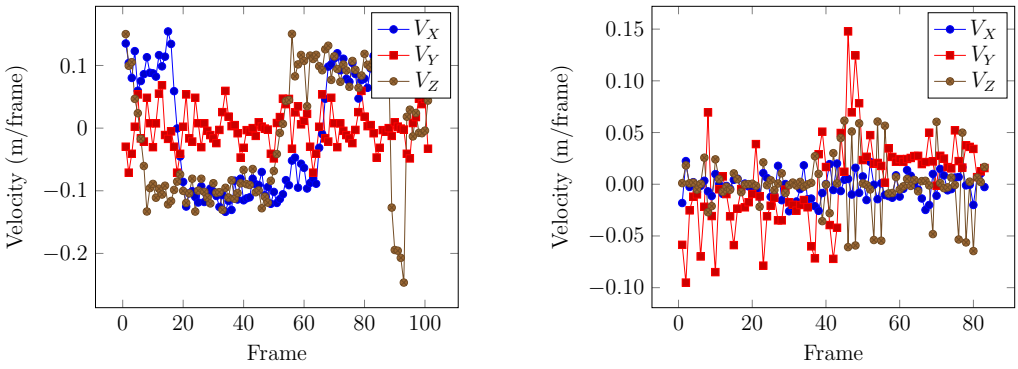
The fig.5.6a, fig.5.7a, fig.5.6b, fig.5.7b shows the x-test and y-test results in another trial. The coordinates change trends are largely same as sec.5.2.3. However, this subject does not move his hands as smooth as the subject in sec.5.2.3, the velocity figure is harder to analysis.

The participant put one hand forward to a start point at $(0m, 0.5m)$, then moved the hand from start point to the right $(0.5m, 0.5m)$. After that, the participant moved his hand back to the left $(-0.4m, 0.5m)$, and finally to the start point, completing the x-test. For the Y-test, the participant put the hand at $(0.1m, 0.75m)$, then lowered the hand down to $(0m, -0.25m)$, and returned to the start point, completing the y-test.



(a) x-test (b) y-test

Figure 5.6: 3D coordinates of the centre of mass of the hand cluster in world coordinates (experiment 2)



(a) x-test (b) y-test

Figure 5.7: Velocity of the centre of mass of the hand cluster in world coordinates (experiment 2)

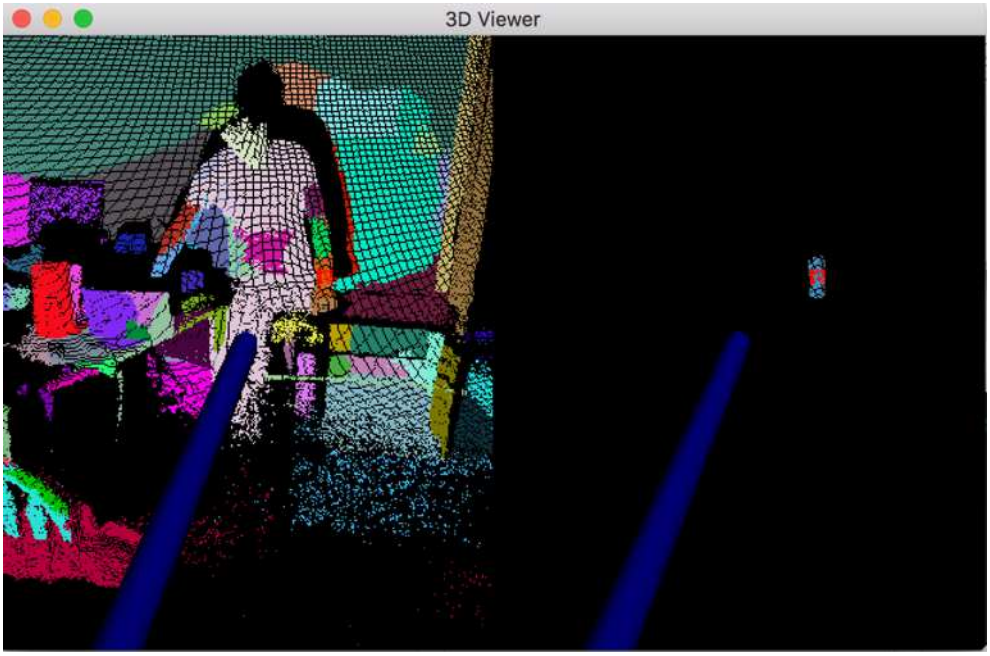


Figure 5.8: A sample frame from the static test. Automated scene segmentation using colour-based region-growing (left). Extracted hand object (right)

Although the velocity figures (fig.5.7a and fig.5.7b) include noise, the overall trends are still recognizable: in fig.5.7a, the velocity in X axis drops from $15m/frame$ to $-0.125m/frame$ at frame 21, which matches the coordination change in fig.5.6a. In fig.5.7b, the overall velocity changes from negative to positive at frame 44, which also matches the coordinates change in fig.5.6b, where the subject move his hand to $-0.25m$.

The other trails also show the same matches between velocity and coordinate changes.

5.2.4 Hand motion modelling and characterization

The centre of mass of the hand across successive frames was used to calculate both velocity and stability of movement. Stability of movement was evaluated by examining the hand velocity in the static test. Results are shown in fig.5.5a. The velocity is very small, and reflects the stable positional data provided by the centre of mass of the hand during the static trial. Fluctuations of this small magnitude are likely contributed by sensor noise, caused by the inaccuracy of the Kinect sensor, and minor segmentation variability. The velocity of the hand was derived from the interpolated times and positions. The velocity was decomposed to the two directions of

motion and shown in fig.5.5b and fig.5.5c separately. The noise in x/y-test velocity figures are significantly larger than that in static test, because it is hard to move the hand stable and slow for test subjects. However, the overall treads of velocity are still unambiguously matched the coordinates changes. Also, the coordinates in fig.5.4a, fig.5.4b and fig.5.4c matches the tape measure placed on the desk for reference.

5.3 Conclusion

This chapter presents work to capture and track hands using a perspective-independent 3D computer vision-based approach in a complex, cluttered, and dynamic indoor environment. We proposed a novel process for tracking and analysing hand movement in these more complex clinic environments. To accomplish this, we processed entirely 3D video data, overcoming two substantial limitations with existing 2D approaches. First, 3D video data from multiple sensors can be combined into a more comprehensive global scene, overcoming issues such as environmental- and self-occlusion, and expanding the field of view of the system. Combining data from multiple sensors in 2D is challenging and often not possible. Second, the resulting global scene is perspective-independent. In real world, clinical environments, the sensors placement is for convenience rather than optimal system performance. Accordingly, we evaluated the performance of several keypoint and feature extraction methods for aligning multiple 3D point clouds with various sensor placements. We identified that ISS keypoints and SHOT features provided the most robust alignment between two point clouds over a range of translations and rotations. Using this alignment method, we then determined that colour-based region-growth segmentation was effective at segmenting a human hand in a complex scene. We used the centre of mass of the segmented hand to track the hands motion over successive frames, and evaluated the hands motion stability and velocity characteristics. We conclude that with this method we can effectively combine the 3D data from multiple, arbitrarily placed depth sensors, segment a human hand, and track the hand over successive frames.

We noted several limitations through the exploratory nature of this study and the experimental

nature of the methodology. Most notably, we evaluated the different combinations of keypoints and features on a relatively small synthetic data set. Manual transformations were performed on a complex test point cloud, creating the synthetic set of target clouds. Future work must extend this evaluation strategy to real-world data captured from multiple sensors in different physical locations. Our proof-of-concept hand tracking also required manual initialization to identify which of the segmented clouds was the hand. A fully-automated process of hand segmentation and identification is critical for clinical applications. Furthermore, if the segmentation failed to identify the hand as a clustered object the tracker may permanently lose the target. This could happen, for example, if the segmentation combined the hand and arm. Once an incorrect cluster is chosen, our current tracker cannot recover without user intervention. We propose utilizing the hand colour and point cloud characteristics (e.g., number of points, shape) to perform template matching among frames in conjunction with centre of mass. Finally, our hand motion modelling currently characterizes motion stability and velocity. We evaluated these motion characteristics through a static test and a simple dynamic test. We plan to expand testing of these characteristics on larger, more diverse and more dynamic test sets. Furthermore, clinicians identified time to target as another clinically relevant characteristic, particularly with respect to the evaluation of tasks such as ultrasound competency. Accordingly, we propose extending the capabilities of our model to include temporal measures.

Notwithstanding the limitations of the current study, our preliminary results suggest that automated multi-sensor registration, hand segmentation, tracking and modelling in full 3D is promising. Our future work will seek to address the limitations of the current study, providing more robust and generalizable results, toward a more automated system. Ultimately, a robust, automated 3D hand tracker that can incorporate the data from multiple arbitrarily placed sensors can support the automated assessment of care delivery. This can, for example, allow better understanding of a clinicians skill during ultrasound training, or ensure the proper execution of surgical procedures. Furthermore, automated systems of 3D tracking can also find broader applications in gaming, human-computer interactions and security.

Chapter 6 Conclusion and future work

This chapter summarizes the work presented in this thesis, including a discussion of the main results and contributions, then outlines the directions of future work. The work described in this thesis outlines the theoretical background for a system that can track an ultrasound technicians hand movements in complex 3D environments using 3D computer vision. However, realizing such a system still requires significant work.

6.1 Conclusion

This thesis outlines the theoretical work toward a high-level framework for an automatic approach of ultrasound operator skill level analysis. Chapter 1 introduces the objectives of my work: 1. Developing a method of automatic ultrasound image capturing and analysis; 2. Extending the effective field of view of a 3D computer vision system using Microsoft Kinect 2 sensors through an automated, calibration free, multi-sensor fusion algorithm; and 3. Developing a semi-automatic method of tracking an ultrasound operators hand in complex indoor environments in 3D space using computer vision.

Chapter 2 reviews the overall background of this research, including ultrasound image analysis, multi-sensor fusion (2D image registration and 3D point cloud registration) and hand tracking (appearance-based and model-based approaches).

Chapter 3 outlines the use of image analysis techniques to analyse ultrasound images, extracting transducer movements from a sequence of ultrasound images. A distributed capturing system was designed to fit the synchronization and high-throughput requirements for capturing RGB-

D data from multiple Microsoft Kinect 2 and ultrasound data from ultrasound machines. The captured data was analysed to find differences in patterns of transducer movements between novice and expert operators, and the results showed that the differences are significant.

Chapter 4 presents the evaluation of all available keypoint detectors and descriptors available in PCL [9] to identify the best combination for pair-wise registration [92]. This optimal pair was used in an automatic, calibration-free multi-sensor fusion algorithm. The results show that ISS performed best among all keypoint detectors. There are multiple keypoint descriptors, such as IntensitySpin, SHOT and SHOTColor, that perform well. Importantly, this chapter contains a discussion about the abnormities observed in the evaluation including potential explanations. For example, the SIFT detector shows linearly degrading symmetrical performance with an additional cyclical modulation, which may cause by the floating-point precision issues on boundaries during voxel down-sampling.

Chapter 5 outlines the examination of existing hand tracking algorithms and proposes a semi-automatic hand tracking algorithm in 3D point clouds. This proposed algorithm is suitable for complex, cluttered, and dynamic indoor environments. Point clouds are first aligned together to generate a more comprehensive global scene, which extended field of view and overcome occlusion issues. Then the colour-based region-growth segmentation is applied to the scene, and segment it into multiple clusters, and further compute the centre of mass of each cluster. The cluster of the hand is manually identified in the first frame, then automatically tracked with nearest cluster algorithm in following frames.

6.2 Future work

This thesis outlines the theoretical development and partial implementation of a framework for automatic analysis of an ultrasound operators skill level. The implementations were focused on multi-sensor fusion and hand tracking, using Microsoft Kinect 2 data and extracting hand movements from ultrasound images. Future work on this project includes:

1. The automated analysis of ultrasound images for all identified key factors, including time-to-acquisition, edges sharpness and position of big structures, with a single metric synthesizing all these results that can be used to represent the ultrasound image quality.

2. Analysis patterns of ultrasound-image-quality-over-time from both novice and expert scans.

   The addition of a sophisticated outlier removal filter before registering point clouds. The depth data of Kinect sensor suffer from noise [98], A sophisticated outlier removal filter could help reducing or suppressing the error rate for point cloud registration.

3. The improvement of existing feature descriptors or the development of a better feature descriptor which is more tolerant to the perspective changes. Most registration failures were caused by inaccurate feature descriptors. This was evident because the keypoints extracted from the point clouds remained stable before and after transformation, while the outputs of the feature descriptors were more sensitive to transformation.

4. Improvement of the point cloud segmentation algorithm. Currently, the segmentation algorithm is sensitive to noise and illumination conditions, because it was based on Euclidean distances in 3D space and RGB colour similarities between two surfaces. Illumination changes may cause the segmentation algorithm to fail, and must be addressed.

5. Automatic identification of the hand cluster in the first frame by template matching or other methods.

Appendix A Distributed recording system

A.1 Controller node

```
1  import wx
2  import sys
3  import cv2
4  import zmq
5  import json
6  import threading
7  import time
8  import struct
9  import numpy
10 import Queue
11 import logging
12
13 logging.basicConfig(level=logging.DEBUG)
14
15
16 class Parser(object):
17         def __init__(self, window, socket):
18                 self.queue = Queue.Queue()
19                 self.is_shutdown = False
20                 self.is_saving = False
```

```
21              self.thread_save = None
22              self.thread_receive = None
23              self.window = window
24              self.socket = socket
25
26      def receive_start(self):
27              self.is_shutdown = False
28              self.is_saving = False
29              self.thread_receive = threading.Thread(
30                      target=self.receive_frame)
31              self.thread_receive.start()
32
33      def receive_stop(self):
34              self.is_shutdown = True
35              self.is_saving = False
36              self.thread_receive.join()
37
38      def save_start(self):
39              self.is_saving = True
40
41      def save_stop(self):
42              self.is_saving = False
43
44      def receive_frame(self):
45              header_s = struct.Struct('13cliiiiii')
46
47              while not self.is_shutdown:
48                      # receive and decode
49                      # logging.debug('waiting...')
```

```
50                  try:
51                      msg = self.socket.recv(flags=zmq.NOBLOCK)
52                  except zmq.Again:
53                      logging.debug('no_data_received')
54                      time.sleep(0.1)
55                      continue
56
57                  offset = 0
58                  header = header_s.unpack(msg[offset:header_s.size])
59                  header = list(header)
60                  # logging.debug(header)
61                  serial = ''.join(header[0:12])
62                  header = header[13:]
63                  offset = offset + header_s.size
64
65                  timestamp = header.pop(0)
66
67                  rgbwidth = header.pop(0)
68                  rgbheight = header.pop(0)
69                  rgbbp = header.pop(0)
70                  rgbsize = rgbwidth * rgbheight * rgbbp
71                  rgb = msg[offset:offset+rgbsize]
72                  offset = offset + rgbsize
73
74                  depthwidth = header.pop(0)
75                  depthheight = header.pop(0)
76                  depthbp = header.pop(0)
77                  depthsize = depthwidth * depthheight * depthbp
78                  depth = msg[offset:offset+depthsize]
```

```
79                  offset = offset + depthsize
80
81                  regsize = depthsize
82                  reg = msg[offset:offset+regsize]
83                  offset = offset + regsize
84                  if offset != len(msg):
85                          logging.warning('redundant data? decoded: %s all: %s',
                                offset, len(msg))
86
87                  rgb = numpy.fromstring(rgb, numpy.uint8)
88                  rgb = numpy.reshape(rgb, (rgbheight, rgbwidth, 4))
89                  depth = numpy.fromstring(depth, numpy.float32)
90                  depth = numpy.reshape(depth, (depthheight, depthwidth))
91                  reg = numpy.fromstring(reg, numpy.uint8)
92                  reg = numpy.reshape(reg, (depthheight, depthwidth, 4))
93
94                  # show image
95                  self.window.updateDevice(serial, rgb, depth, reg)
96
97                  if self.is_saving and not self.is_shutdown:
98                          try:
99                                  self.queue.put((serial, timestamp, rgb, depth, reg))
100                         except Queue.Full:
101                                 logging.error('Queue full!! skip current frame')
102                                 continue
103
104                         if not self.thread_save:
105                                 thread_save = threading.Thread(target=self.save)
106                                 thread_save.start()
```

```
107                                     self.thread_save = thread_save
108
109                     if self.is_shutdown:
110                             # shutdown
111                             thread_save.join()
112
113                     # socket.send_string(command)
114                     # # always reset command
115                     # if command != 'ACK':
116                     #  command = 'ACK'
117             logging.info('mainloop: exited')
118
119     def save(self):
120             while True:
121                     try:
122                             item = self.queue.get_nowait()
123                     except Queue.Empty:
124                             if not self.is_saving:
125                                     break
126                             logging.debug('save: queue underflow')
127                             time.sleep(1)
128                             continue
129
130                     serial = item[0]
131                     timestamp = item[1]
132                     logging.debug('saving %s %s', serial, timestamp)
133                     numpy.save('%s-%s-rgb.npy' % (serial, timestamp), item[2])
134                     numpy.save('%s-%s-depth.npy' % (serial, timestamp), item[3])
135                     numpy.save('%s-%s-reg.npy' % (serial, timestamp), item[4])
```

```
136
137                 self.thread_save = None
138                 logging.info('save:_exited')
139
140
141 class MainWindow(wx.Frame):
142         def __init__(self, parent, title, size):
143                 super(MainWindow, self).__init__(parent, title=title, size=size)
144
145                 self.InitUI()
146                 self.Centre()
147                 self.InitNet()
148                 self.Show()
149
150                 self.devices = {}
151                 self.parser = Parser(self, self.socket)
152                 self.parser.receive_start()
153
154         def InitUI(self):
155                 self.Bind(wx.EVT_CLOSE, self.onClose)
156
157                 panel = wx.Panel(self, -1)
158                 self.panel = panel
159                 vbox = wx.BoxSizer(wx.VERTICAL)
160                 self.vbox = vbox
161
162                 hbox = wx.BoxSizer(wx.HORIZONTAL)
163                 buttonStart = wx.Button(panel, label='Start', size=(70, 30))
164                 buttonStart.Bind(wx.EVT_BUTTON, self.onStart)
```

```
165             hbox.Add(buttonStart, border=10)
166             buttonStop = wx.Button(panel, label='Stop', size=(70, 30))
167             buttonStop.Bind(wx.EVT_BUTTON, self.onStop)
168             hbox.Add(buttonStop, border=10)
169             vbox.Add(hbox)
170
171             panel.SetSizer(vbox)
172
173     def InitNet(self):
174             fp = open('config.json')
175             self.config = json.load(fp)
176
177             self.ctx = zmq.Context()
178             self.socket = self.ctx.socket(zmq.SUB)
179             self.socket.setsockopt(zmq.SUBSCRIBE, '')
180             for url in self.config['collectors']:
181                     print 'subscribe_to', url
182                     self.socket.connect(url)
183
184
185     def onStart(self, e):
186             print 'start'
187             self.parser.save_start()
188
189     def onStop(self, e):
190             print 'stop'
191             self.parser.save_stop()
192
193     def onClose(self, e):
```

```
194             print 'close'
195             self.parser.receive_stop()
196             self.Destroy()
197
198     def updateDevice(self, serial, rgb, depth, reg):
199             if serial not in self.devices:
200                     hbox = wx.BoxSizer(wx.HORIZONTAL)
201                     # create
202                     self.devices[serial] = {
203                             'rgb': wx.StaticBitmap(self.panel),
204                             'depth': wx.StaticBitmap(self.panel),
205                             'reg': wx.StaticBitmap(self.panel)
206                     }
207                     hbox.Add(self.devices[serial]['rgb'], border=10)
208                     hbox.Add(self.devices[serial]['depth'], border=10)
209                     hbox.Add(self.devices[serial]['reg'], border=10)
210                     self.vbox.Add(hbox, border=10)
211
212             # update
213             rgb = cv2.resize(rgb, (256, 256))
214             rgb = cv2.cvtColor(rgb, cv2.COLOR_BGRA2RGB)
215             depth = cv2.resize(depth, (256, 256))
216             # depth_vis = cv2.cvtColor(depth, cv2.COLOR_BGRA2RGB)
217             reg = cv2.resize(reg, (256, 256))
218             reg = cv2.cvtColor(reg, cv2.COLOR_BGRA2RGB)
219             self.updateImage(self.devices[serial]['rgb'], rgb)
220             # self.updateImage(self.devices[serial]['depth'], depth_vis)
221             self.updateImage(self.devices[serial]['reg'], reg)
222
```

```
223         def updateImage(self, wximage, img):
224                 wximg = wx.EmptyImage(img.shape[1], img.shape[0])
225                 wximg.SetData(img.tostring())
226                 wxbitmap = wximg.ConvertToBitmap()
227                 wximage.SetBitmap(wxbitmap)
228
229
230 if __name__ == '__main__':
231         app = wx.App(False)
232         MainWindow(None, "Command_Centre", (260, 180))
233         app.MainLoop()
```

A.2 Recording node

```
1  #include <iostream>
2  #include <opencv2/opencv.hpp>
3  #include <libfreenect2/libfreenect2.hpp>
4  #include <libfreenect2/frame_listener_impl.h>
5  #include <libfreenect2/packet_pipeline.h>
6  #include <libfreenect2/registration.h>
7  #include <zmq.h>
8  #include <sys/time.h>
9
10
11 extern int errno;
12
13 struct header_t{
14         char serial[13];
```

```
15          long timestamp;
16
17          int rgb_width;
18          int rgb_height;
19          int rgb_bytes_per_pixel;
20
21          int depth_width;
22          int depth_height;
23          int depth_bytes_per_pixel;
24  };
25
26  bool protonect_shutdown = false;
27  bool big_image = false;
28
29  int main(int argc, char **argv){
30          libfreenect2::Freenect2 freenect2;
31          if((argc != 3) && (argc != 4)){
32                  std::cout << "usage:" <<
33                          argv[0] << " bindURL_serial" << std::endl <<
34                          "example: " << argv[0] << " tcp://*:5555_123456"
35                          << std::endl;
36
37                  int kinectnum = freenect2.enumerateDevices();
38                  std::cout << "detected_Kinects: " <<
39                          kinectnum << std::endl;
40                  for(int i = 0; i < kinectnum; i++){
41                          std::cout << freenect2.openDevice(
42                                  i, new libfreenect2::CpuPacketPipeline()
43                                  )->getSerialNumber() << std::endl;
```

```
44              }
45
46              return 0;
47          }
48
49          char *bindurl = argv[1];
50          char *kinectserial = argv[2];
51          if((argc == 4) && (std::string(argv[3]) == "true")){
52              big_image = true;
53              std::cout << "big_image_model" << std::endl;
54          }
55          std::cout << "small_image_model" << std::endl;
56
57          void *zcontext = zmq_ctx_new();
58          void *socket = zmq_socket(zcontext, ZMQ_PUB);
59          std::cout << "binding_" << bindurl << std::endl;
60          if(zmq_bind(socket, bindurl) != 0){
61              std::cerr << "bind_failed" << std::endl;
62              return 1;
63          }
64          std::cout << "binded" << std::endl;
65
66          libfreenect2::PacketPipeline *pipeline = NULL;
67          // pipeline = new libfreenect2::CudaPacketPipeline();
68          pipeline = new libfreenect2::OpenCLPacketPipeline();
69          // pipeline = new libfreenect2::OpenGLPacketPipeline();
70          // pipeline = new libfreenect2::CpuPacketPipeline();
71
72          libfreenect2::Freenect2Device *dev = freenect2.openDevice(
```

```
73              kinectserial, pipeline
74              );
75          if(dev == NULL){
76              std::cerr << "unable to open device." << std::endl;
77              return 1;
78          }
79          std::cout << "device opened" << std::endl;
80
81          libfreenect2::SyncMultiFrameListener listener(
82                  libfreenect2::Frame::Color | libfreenect2::Frame::Depth);
83          dev->setColorFrameListener(&listener);
84          dev->setIrAndDepthFrameListener(&listener);
85          dev->start();
86
87          std::cout << "device serial: " << dev->getSerialNumber() << std::endl;
88          std::cout << "device firmware: " << dev->getFirmwareVersion() << std::endl;
89
90          libfreenect2::Registration* registration =
91              new libfreenect2::Registration(
92                  dev->getIrCameraParams(), dev->getColorCameraParams());
93
94          libfreenect2::FrameMap frame;
95          while(!protonect_shutdown){
96              listener.waitForNewFrame(frame);
97              libfreenect2::Frame *rgb = frame[libfreenect2::Frame::Color];
98              libfreenect2::Frame *depth = frame[libfreenect2::Frame::Depth];
99              libfreenect2::Frame undistorted(depth->width, depth->height,
100                 depth->bytes_per_pixel);
101             libfreenect2::Frame registered(depth->width, depth->height,
```

```
102                     depth->bytes_per_pixel);
103             registration->apply(rgb, depth, &undistorted, &registered);
104
105
106             header_t header;
107             memcpy(header.serial, kinectserial, strlen(kinectserial));
108             struct timeval tp;
109             gettimeofday(&tp, NULL);
110             header.timestamp = tp.tv_sec * 1000 + tp.tv_usec / 1000;
111             cv::Mat rgbsmall;
112             if(big_image == true){
113                     header.rgb_width = rgb->width;
114                     header.rgb_height = rgb->height;
115                     header.rgb_bytes_per_pixel = rgb->bytes_per_pixel;
116             }
117             else{
118                     cv::Mat rgbmat(rgb->height, rgb->width, CV_8UC4, rgb->data);
119                     cv::resize(rgbmat, rgbsmall,
120                             cv::Size(depth->width, depth->height));
121                     header.rgb_width = rgbsmall.cols;
122                     header.rgb_height = rgbsmall.rows;
123                     header.rgb_bytes_per_pixel = 4;
124             }
125
126             header.depth_width = depth->width;
127             header.depth_height = depth->height;
128             header.depth_bytes_per_pixel = depth->bytes_per_pixel;
129
130             size_t rgbsize = 0;
```

```
131         if(big_image == true){
132             rgbsize = rgb->width * rgb->height * rgb->bytes_per_pixel;
133         }
134         else{
135             rgbsize = rgbsmall.cols * rgbsmall.rows * 4;
136         }
137         size_t depthsize = depth->width * depth->height *
138             depth->bytes_per_pixel;
139         size_t regsize = registered.width * registered.height *
140             registered.bytes_per_pixel;
141
142         zmq_msg_t request;
143         zmq_msg_init_size(&request, sizeof(header) +
144             rgbsize + depthsize + regsize
145             );
146         char *pdata = (char*)zmq_msg_data(&request);
147         memcpy(pdata, &header, sizeof(header));
148         pdata += sizeof(header);
149         if(big_image == true){
150             memcpy(pdata, rgb->data, rgbsize);
151         }
152         else{
153             memcpy(pdata, rgbsmall.data, rgbsize);
154         }
155         pdata += rgbsize;
156         memcpy(pdata, depth->data, depthsize);
157         pdata += depthsize;
158         memcpy(pdata, registered.data, regsize);
159         if(zmq_sendmsg(socket, &request, 0) < 0){
```

```
160                         std::cerr << "send_failed" << errno << std::endl;
161                 }
162                 std::cout << "Sent" << std::endl;
163
164                 listener.release(frame);
165         }
166
167         delete registration;
168         dev->stop();
169         dev->close();
170
171         zmq_close(socket);
172         zmq_ctx_destroy(zcontext);
173
174         return 0;
175 }
```

Bibliography

[1] M. Camplani and L. Salgado, "Efficient spatio-temporal hole filling strategy for kinect depth maps," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2012, pp. 82 900E–82 900E–10.

[2] E. Piette, R. Daoust, and A. Denault, "Basic concepts in the use of thoracic and lung ultrasound," *Current Opinion in Anesthesiology*, vol. 26, no. 1, pp. 20–30, 2013.

[3] L. Zieleskiewicz, L. Muller, K. Lakhal, Z. Meresse, C. Arbelot, P.-M. Bertrand, B. Bouhemad, B. Cholley, D. Demory, and S. Duperret, "Point-of-care ultrasound in intensive care units: assessment of 1073 procedures in a multicentric, prospective, observational study," *Intensive care medicine*, vol. 41, no. 9, pp. 1638–1647, 2015.

[4] M. D. Lo, S. H. Ackley, and P. Solari, "Homemade ultrasound phantom for teaching identification of superficial soft tissue abscess," *Emergency Medicine Journal*, vol. 29, no. 9, pp. 738–741, 2012.

[5] B. S. Hertzberg, M. A. Kliewer, J. D. Bowie, B. A. Carroll, D. H. DeLong, L. Gray, and R. C. Nelson, "Physician training requirements in sonography: how many cases are needed for competence?" *American Journal of Roentgenology*, vol. 174, no. 5, pp. 1221–1227, 2000.

[6] C. J. Gardner, S. Brown, S. Hagen-Ansert, P. Harrigan, J. Kisslo, K. Kisslo, O. L. Kwan, F. Menapace, C. Otto, and N. Pandian, "Guidelines for cardiac sonographer education: report of the american society of echocardiography sonographer education and training committee," *Journal of the American Society of Echocardiography*, vol. 5, no. 6, pp. 635–639, 1992.

[7] Z. Chen, M. S. Shehata, M. Gong, H. Carnahan, A. Dubrowski, and A. Smith, "Feasibility of a semi-automated approach to grading point of care ultrasound image generation skills," in *Image and Vision Computing New Zealand (IVCNZ), 2015 International Conference on*. IEEE, 2015, pp. 1–5.

[8] Z. Chen, S. Czarnuch, A. Smith, and M. Shehata, "Performance evaluation of 3d keypoints and descriptors," in *International Symposium on Visual Computing*. Springer, 2016, pp. 410–420.

[9] PCL, "Point cloud library," 2016.

[10] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," pp. 1817–1824, 2011.

[11] A. Dubrowski, R. Sidhu, J. Park, and H. Carnahan, "Quantification of motion characteristics and forces applied to tissues during suturing," *The American journal of surgery*, vol. 190, no. 1, pp. 131–136, 2005.

[12] C. Prinz, J. Dohrmann, F. van Buuren, T. Bitter, N. Bogunovic, D. Horstkotte, and L. Faber, "The importance of training in echocardiography: a validation study using pocket echocardiography," *Journal of Cardiovascular Medicine*, vol. 13, no. 11, pp. 700–707, 2012.

[13] M. C. Corretti, T. J. Anderson, E. J. Benjamin, D. Celermajer, F. Charbonneau, M. A. Creager, J. Deanfield, H. Drexler, M. Gerhard-Herman, D. Herrington *et al.*, "Guidelines for the ultrasound assessment of endothelial-dependent flow-mediated vasodilation of the brachial artery," *Journal of the American College of Cardiology*, vol. 39, pp. 257–265, 2002.

[14] H. B. Hammer, P. Bolton-King, V. Bakkeheim, T. H. Berg, E. Sundt, A. K. Kongtorp, and E. A. Haavardsholm, "Examination of intra and interrater reliability with a new ultrasonographic reference atlas for scoring of synovitis in patients with rheumatoid arthritis," *Annals of the rheumatic diseases*, p. annrheumdis152926, 2011.

[15] B. J. Kimura, M. Bocchicchio, C. L. Willis, and A. N. DeMaria, "Screening cardiac ultrasonographic examination in patients with suspected cardiac disease in the emergency department," *American heart journal*, vol. 142, no. 2, pp. 324–330, 2001.

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

[17] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of psnr in image/video quality assessment," *Electronics letters*, vol. 44, no. 13, pp. 800–801, 2008.

[18] J. F. Krcker, C. R. Meyer, G. L. LeCarpentier, J. B. Fowlkes, and P. L. Carson, "3d spatial compounding of ultrasound images using image-based nonrigid registration," *Ultrasound in medicine & biology*, vol. 26, no. 9, pp. 1475–1488, 2000.

[19] C. ø. Simpson, "Objective image quality metrics for ultrasound imaging," 2009.

[20] T. Venkat and N. Rao, "Assessment of diverse quality metrics for medical images including mammography," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 14, no. 11, p. 56, 2014.

[21] M. C. Hemmsen, M. M. Petersen, S. I. Nikolov, M. B. Nielsen, and J. A. Jensen, "Ultrasound image quality assessment: A framework for evaluation of clinical image quality," in *SPIE Medical Imaging*. International Society for Optics and Photonics, 2010, pp. 76 290C–76 290C–12.

[22] T. J. MacGillivray, E. Ross, H. A. Simpson, and C. A. Greig, "3d freehand ultrasound for in vivo determination of human skeletal muscle volume," *Ultrasound in medicine & biology*, vol. 35, no. 6, pp. 928–935, 2009.

[23] A. Krupa, G. Fichtinger, and G. D. Hager, "Full motion tracking in ultrasound using image speckle information and visual servoing," pp. 2458–2464, 2007.

[24] C. Laporte and T. Arbel, "Combinatorial and probabilistic fusion of noisy correlation measurements for untracked freehand 3-d ultrasound," *Medical Imaging, IEEE Transactions on*, vol. 27, no. 7, pp. 984–994, 2008.

[25] ——, "Learning to estimate out-of-plane motion in ultrasound imagery of real tissue," *Medical image analysis*, vol. 15, no. 2, pp. 202–213, 2011.

[26] B. Tordoff and D. W. Murray, "Guided sampling and consensus for motion estimation," pp. 82–96, 2002.

[27] K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, and M. Ouhyoung, "Video stabilization using robust feature trajectories," pp. 1397–1404, 2009.

[28] D. P. Bahner, E. J. Adkins, R. Nagel, D. Way, H. A. Werman, and N. A. Royall, "Brightness mode quality ultrasound imaging examination technique (b-quiet) quantifying quality in ultrasound imaging," *Journal of Ultrasound in Medicine*, vol. 30, no. 12, pp. 1649–1655, 2011.

[29] I.-B. A. I. Registration, "http://www.mathworks.com/help/images/intensity-based-automatic-image-registration.html," 2015.

[30] T. Heap and D. Hogg, "Towards 3d hand tracking using a deformable model," in *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on.* IEEE, 1996, pp. 140–145.

[31] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect."

[32] M. Bray, E. Koller-Meier, P. Mller, L. Van Gool, and N. N. Schraudolph, "3d hand tracking by rapid stochastic gradient descent using a skinning model," in *In 1st European Conference on Visual Media Production (CVMP.* Citeseer, 2004.

[33] R. P. Poudel, "3d hand tracking." Ph.D. dissertation, Bournemouth University, 2014.

[34] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *BmVC*, vol. 1, 2011, p. 3.

[35] M. Donoser and H. Bischof, "Real time appearance based hand tracking," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on.* IEEE, 2008, pp. 1–4.

[36] Y. Cui and J. Weng, "Appearance-based hand sign recognition from intensity image sequences," *Computer Vision and Image Understanding*, vol. 78, no. 2, pp. 157–176, 2000.

[37] B. Stenger, A. Thayananthan, P. H. Torr, and R. Cipolla, "Model-based hand tracking using a hierarchical bayesian filter," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 9, pp. 1372–1384, 2006.

[38] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013.

[39] L. Motion, "Api overview," 2016.

[40] A. Baak, M. Mller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 71–98.

[41] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 188, 2012.

[42] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab, "Human skeleton tracking from depth data using geodesic distances and optical flow," *Image and Vision Computing*, vol. 30, no. 3, pp. 217–226, 2012.

[43] C. Keskin, F. Kra, Y. E. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 119–137.

[44] J. L. Raheja, A. Chaudhary, and K. Singal, "Tracking of fingertips and centers of palm using kinect," in *Computational intelligence, modelling and simulation (CIMSiM), 2011 third international conference on*. IEEE, 2011, pp. 248–252.

[45] L. G. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992.

[46] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981.

[47] Microsoft, "Kinect for windows sdk," 2017.

[48] I. Occipital, "Using your structure sensor for the first time," 2017.

[49] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas, "Registration of point cloud data from a geometric optimization perspective," pp. 22–31, 2004.

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

[51] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi, "Large-scale and drift-free surface reconstruction using online subvolume registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4475–4483.

[52] R. Hnsch, T. Weber, and O. Hellwich, "Comparison of 3d interest point detectors and descriptors for point cloud fusion," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 3, p. 57, 2014.

[53] I. Sipiran, B. Bustos, I. Sipiran, and B. Bustos, "Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes," 2011.

[54] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," pp. 689–696, 2009.

[55] S. Filipe and L. A. Alexandre, "A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset," pp. 476–483, 2014.

[56] S. M. Smith and J. M. Brady, "Susana new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[57] C. Harris and M. Stephens, "A combined corner and edge detector." p. 50, 1988.

[58] D. G. Lowe, "Object recognition from local scale-invariant features," pp. 1150–1157, 1999.

[59] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Narf: 3d range image features for object recognition," 2010.

[60] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," pp. 183–196, 2010.

[61] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," pp. 2548–2555, 2011.

[62] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," pp. 3212–3217, 2009.

[63] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," pp. 2155–2162, 2010.

[64] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "Cad-model recognition and 6dof pose estimation using 3d cues," pp. 585–592, 2011.

[65] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1265–1278, 2005.

[66] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," pp. 356–369, 2010.

[67] F. Tombari, S. Salti, and L. D. Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching," pp. 809–812, 2011.

[68] L. A. Alexandre, "3d descriptors for object and category recognition: a comparative evaluation," p. 7, 2012.

[69] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," pp. 3384–3391, 2008.

[70] P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3d objects," *International Journal of Computer Vision*, vol. 73, no. 3, pp. 263–284, 2007.

[71] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *Computer Vision and*

Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009, pp. 1746–1753.

[72] D. S. Alexiadis, P. Kelly, P. Daras, N. E. O'Connor, T. Boubekeur, and M. B. Moussa, "Evaluating a dancer's performance using kinect-based skeleton tracking," in *Proceedings of the 19th ACM international conference on Multimedia.* ACM, 2011, pp. 659–662.

[73] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[74] A. Hernndez-Vela, N. Zlateva, A. Marinov, M. Reyes, P. Radeva, D. Dimov, and S. Escalera, "Graph cuts optimization for multi-limb human segmentation in depth maps," pp. 726–732, 2012.

[75] S. Czarnuch and A. Mihailidis, "Development and evaluation of a hand tracker using depth images captured from an overhead perspective," *Disability and Rehabilitation: Assistive Technology*, pp. 1–8, 2015.

[76] C. Li and K. Kitani, "Pixel-level hand detection in ego-centric videos," pp. 3570–3577, 2013.

[77] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," pp. 1106–1113, 2014.

[78] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt, "Fast and robust hand tracking using detection-guided optimization," pp. 3213–3221, 2015.

[79] Z. Ma and E. Wu, "Real-time and robust hand tracking with a single depth camera," *The Visual Computer*, vol. 30, no. 10, pp. 1133–1144, 2014.

[80] E. S. Inc., "Dvi2usb 3.0 - usb video grabber for hdmi video capture," 2017.

[81] M. J. Ledesma-Carbayo, J. Kybic, M. Desco, A. Santos, M. Suhling, P. Hunziker, and M. Unser, "Spatio-temporal nonrigid registration for ultrasound cardiac motion estimation," *IEEE transactions on medical imaging*, vol. 24, no. 9, pp. 1113–1126, 2005.

[82] F. Yeung, S. F. Levinson, D. Fu, and K. J. Parker, "Feature-adaptive motion tracking of ultrasound image sequences using a deformable mesh," *IEEE Transactions on Medical Imaging*, vol. 17, no. 6, pp. 945–956, 1998.

[83] B. Kim, J. L. Boes, P. H. Bland, T. L. Chenevert, and C. R. Meyer, "Motion correction in fmri via registration of individual slices into an anatomical volume," 1999.

[84] Microsoft, "Kinect hardware," 2017.

[85] ——, "Known issues with the kinect for windows v2 sensor," 2017.

[86] A. Inc., "Os x el capitan: Set the date and time on your mac," 2017.

[87] T. M. Scalea, A. Rodriguez, W. C. Chiu, F. D. Brenneman, W. F. Fallon, K. Kato, M. G. McKenney, M. L. Nerlich, M. G. Ochsner, and H. Yoshii, "Focused assessment with sonography for trauma (fast): results from an international consensus conference," *Journal of Trauma and Acute Care Surgery*, vol. 46, no. 3, pp. 466–472, 1999.

[88] ffmpeg, "Ffmpeg," 2017.

[89] D. T. Solutions, "Sports video performance analysis software — gamebreaker sportscode," 2015.

[90] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 891–923, 1998.

[91] I. Van Der Bom, S. Klein, M. Staring, R. Homan, L. Bartels, and J. Pluim, "Evaluation of optimization methods for intensity-based 2d-3d registration in x-ray guided interventions," in *Proc. SPIE*, vol. 7962, 2011, pp. 796 223–796 223–15.

[92] PointClouds.org, "The pcl registration api," 2017.

[93] I. The MathWorks, "3-d point cloud registration and stitching," 2017.

[94] F. Tombari, "Keypoints and features," 2013.

[95] Q. Zhan, Y. Liang, and Y. Xiao, "Color-based segmentation of point clouds," in *Proc. ISPRS Laser Scan. Workshop*, 2009, pp. 248–252.

[96] A. Tremeau and N. Borel, "A region growing and merging algorithm to color segmentation," *Pattern recognition*, vol. 30, no. 7, pp. 1191–1203, 1997.

[97] PointClouds.org, "Region growing segmentation," 2016.

[98] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on.* IEEE, 2012, pp. 524–530.