STEREO CALIBRATION OF DEPTH SENSORS WITH 3D CORRESPONDENCES AND SMOOTH INTERPOLANTS

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Chrysanthi Chaleva Ntina

May 2013

STEREO CALIBRATION OF DEPTH SENSORS WITH 3D CORRESPONDENCES AND SMOOTH INTERPOLANTS

Chrysanthi Chaleva Ntina

APPROVED:

Dr. Zhigang Deng, Chairman
Dept. of Computer Science

Dr. Guoning Chen
Dept. of Computer Science

Dr. Mina Dawood
Dept. of Civil and Environmental Engineering

Dean, College of Natural Sciences and Mathematics

STEREO CALIBRATION OF DEPTH SENSORS WITH 3D CORRESPONDENCES AND SMOOTH INTERPOLANTS

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Chrysanthi Chaleva Ntina

May 2013

Abstract

The Microsoft Kinect is a novel sensor that besides color images, also returns the actual distance of a captured scene from the camera. Its depth sensing capabilities, along with its affordable, commercial-type availability led to its quick adaptation for research and applications in Computer Vision and Graphics. Recently, multi-Kinect systems are being introduced in order to tackle problems like body scanning, scene reconstruction, and object detection. Multiple-cameras configurations however, must first be calibrated on a common coordinate system, i.e. the relative position of each camera needs to be estimated with respect to a global origin. Up to now, this has been addressed by applying well-established calibration methods, developed for conventional cameras. Such approaches do not take advantage of the additional depth information, and disregard the quantization error model introduced by the depth resolution specifications of the sensor. We propose a novel algorithm for calibrating a pair of depth sensors, based on a recovered affine transformation from very few 3D point correspondences, refined under a non-rigid registration, that accounts for the non-linear sensor acquisition error. The result is a closed form mapping, of the smooth warping type, that compensates for pairwise calibration point differences. The formulation is further complemented by proposing two different ways of efficiently capturing candidate calibration points. Qualitative 3D registration results show significant improvement over the conventional rigid calibration method, and highlight the potential for advanced and more accurate multi-sensor configurations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we state the research topic and motivation of this thesis, and provide an outline of the contents.

1.1 Motivation

The release of the Microsoft Kinect sensor in 2010 has given a new boost to Computer Vision and Graphics research and applications. With its depth sensing capabilities, a new source of scene data, and its affordable, commercial-type availability, it provides a fast and convenient way of enhancing camera setups. The Kinect is a novel sensing device, that apart from being a conventional RGB camera, it also incorporates a depth sensor that can provide the depth value of each frame pixel in the form of distance from the camera in mm units. In the relatively few years that have elapsed since its release, a large body of work has emerged in the literature in

diverse fields, related to using, configuring or expanding the new framework.

At an algorithmic level, Kinect data have been used with conventional image processing methods for human detection [77], person tracking [45], [37], and object detection [65]. Graphics applications include face [82], [32], body [20], [18], [69], [74], and shape [21], [19] scanning, as well as markerless motion capture [8] and scene reconstruction [35], [75]. A range of additional fields have also incorporated the versatility of the Kinect data for solving traditional or newly emerging problems, besides the originally intended gaming and motion-based interfaces. Examples include, Robotics, Human-Computer Interaction (HCI), Animation, Smart(er) Interfaces, and more.

Applications in HCI range from new interactive designs [58], [55], [81], [42], [58], [76] to immersive and virtual reality [72]. In robotics, the sensor's portability and high frame rate [61] has been employed, e.g. by mounting Kinects on mobile robots for visual-based navigation, obstacle avoidance [52], [9], and indoor scene modeling [31], [70]. Moreover, specialized application domains have emerged by incorporating the use of Kinect and replacing traditional cameras. For example, in healthcare applications, Kinect sensors have been used for patient rehabilitation [44], [16]. The above being indicative applications, the diversity of the domains is dictated by the wide range of problems involving shape, scene, and motion measurements in the three-dimensional space.

Many of the associated applications can benefit if extended to capture, process and fuse data from multi-camera setups. In that respect, multi-Kinect systems and setups have been recently introduced in order to increase the 3D spatial coverage

of scanning and reconstruction of a single sensor setup. For example, scanning applications with multiple sensors can capture the subject simultaneously from many different views for 360° body or 180° facial scans. Moreover, hybrid setups combine high-resolution digital cameras with one or multiple Kinect in order to get the best of both worlds (shape and appearance).

A multi-sensor configuration, similar to conventional multi-camera setups, must first be calibrated on a common coordinate system, i.e. the relative position of each camera, and in extension each separate data capture, needs to be estimated with respect to a global origin. The problem of *external or extrinsic sensor calibration* involves estimating a transformation or mapping from each sensor to a single one, selected by convention. The transformations are obtained by solving a pair-wise or global optimization problem.

A prerequisite to solving this inverse, multi-parameter problem is to establish point correspondences among the different views. Such points, referred to as calibration points, attain the role of reference locations in space, whose depiction in the acquired depth images are precisely known or can be specified in advance. In conventional camera calibration, the usual way is finding common 2D points through their underlying image features (i.e. corners, edges, junctions, descriptors) and using their correspondence to solve a linear system. The resulting camera-to-camera transformations are affine maps, that relate the two coordinate systems through rotation and translation.

1.2 Purpose and Aim

In the state-of-the-art Kinect literature, the correspondence and calibration problem have been addressed by applying existing and well-established calibration methods, originally developed for conventional cameras. Such approaches however, do not explicitly take advantage of the additional 3D depth information provided by the sensor and use 2D points from the depth images instead. As a result, they require a large number of images in order to approximate the calibration transformations, without at the same time attaining high accuracy, in terms of the composite views. Furthermore, they disregard the quantization error model introduced by the depth resolution specifications of the sensor.

In this thesis we aim to address these two different but complementary problems, namely a) establishing point correspondences for calibration points using the 3D data and b) proposing a more plausible and natural calibration for the Kinect. Our goal is to achieve efficient, flexible, and highly-accurate stereo pair depth sensor calibration. Efficiency and flexibility relate to the number of required points and the relative locations of the sensors. Accuracy relates to building on the depth information, readily available, and accounting for any deviations in the affinity assumption model, but applying a non-affine mapping refinement.

Towards that end, we use the actual, back-projected 3D positions of detected points in order to estimate a global transformation. This transformation is further refined using a Thin Plate Splines formulation, that functions as a non-rigid registration and can compensate for any errors introduced by non-linear sensor acquisition.

The result of this second step is a closed form mapping, of the smooth warping type, that can compensate for the difference in mismatch error among calibration pairs. The framework is further complemented by proposing two different ways of identifying and acquiring potential calibration points.

As a side note, we will use the terms *Kinect*, *sensor*, and *camera* interchangeably in the rest of this thesis. We will explicitly differentiate between the sensor as a whole and one of its camera components depending on the context of the presentation.

1.3 Challenges

A number of challenges associated with this work relate to the area still being relatively new and the number of free parameters and configuration details. These include:

- A multitude of parameters involved in the physical set-up such as: sensor individuality and errors; capture angles; distance from target; scale of target objects; multi-sensor synchronization; depth-resolution etc.
- The case of extreme angle between the sensors, where the overlapping regions between two views are limited (application of post-registration refinement methods is not possible).
- The lack of ground truth data for quantitative evaluations, dictates qualitative (visual) evaluations and application-related validation (future work).

- The lack of established evaluation protocols or annotated, databases for reference.

1.4 Contributions

The contributions of this thesis are summarized as follows:

- A novel method for depth sensor stereo calibration using point correspondences in 3D space.
- A refinement of the affine registration step using smooth interpolant functions.
- Two proposed methods for obtaining and establishing correspondence in 3D space.
- An overall framework for multiple Kinect-based, simultaneous captures for data registration with possible extensions to more than two sensors and consecutive frames.

1.5 Outline

The remainder of this thesis is organized as follows: Chapter 2 provides a literature overview of works that use multiple depth sensors and the way they approach system calibration. In Chapter 3 we describe in detail the Microsoft Kinect sensor and enumerate the types of depth errors that motivate the proposed calibration

method. In Chapter 4 we justify why the conventional calibration methods are not sufficient and present the proposed modifications. The calibration method is based on finding accurate calibration points, for which we describe two options in Chapter 5. Chapter 6 presents qualitative comparisons and application of the method in examples of two-view 3D object reconstruction. Finally, in Chapter 7 we conclude with a discussion of the method and results and propose topics and directions for further research.

Chapter 2

Related Work

Calibrating the relative position of Kinects in a multi-camera configuration is not a straightforward process that can be done using one unique method. The Kinect can output 3 streams; the color camera produces RGB frames, while the infrared (IR) camera can output both infrared grayscale images as well as (grayscale-coded) depth frames. As a result, one could use one, or a combination of these streams in order to find correspondences and perform calibration. For example, we could choose the color camera of the Kinect as the reference and use the RGB frames for calibration, or alternatively chose the IR camera and thus use the infrared or depth frames. Independently of the stream chosen however, most works with multiple Kinects perform a generic calibration method targeted for conventional cameras. In this chapter, we review recent works that use multiple depth sensors in their setup and describe the approach they follow for their system calibration. We also differentiate between the actual algorithm and the way the calibration points are captured, which

we review in the second part. We present both stereo and multi-stereo calibration methods, since although the proposed method was developed primarily for a stereo sensor pair, it can be generalized and extended to multiple Kinects.

2.1 Multi-Kinect Calibration Approaches

Depending on the number of cameras present in the system and the characteristics of particular setups, the approach usually used for calibration may vary. In this Section we present the most common approaches and outline their limitations.

2.1.1 Conventional Pairwise Stereo Calibration

According to classic theory of multi-view geometry estimation [29, 80], a standard method for conventional stereo calibration is based on estimating the transformation between a camera pair, through a set of point correspondences between world and image plane points. These are established using easily detected or specified points in both 3D and 2D domains, e.g. using calibration objects (checkerboard patterns) and automatic feature detection, markers, or manually specified points. Roughly speaking, the method using multiple image from both views, estimates the relative rotation and translation of one camera to the other (Sec. 3.2.3), with the option of a full calibration in individual intrinsic and extrinsic parameters.

The method is frequently used as a reference and black-box solution in the literature, has been implemented as a MATLAB Calibration Toolbox by Bouguet [12],

and is part of the standard OpenCV (Open Computer Vision) library [2]. A very popular Kinect Toolbox, RGBDemo [13] also offers an option for automatic stereo Kinect calibration. It provides an interface from which a user can capture points and employs the OpenCV implementation.

Berger et al. [8] evaluate setups with a varying number of Kinects (one to four) for markerless motion capture. They use four cameras in the corners of a room and rely on the depth image. Since not many details are provided, we assume that the cameras are calibrated pairwise with no global refinement step. In a subsequent paper, Berger et al. [7] use three Kinect sensors placed in 45 degree angles in order to reconstruct gas flows around occluders. For this setup, which can be seen in Fig. 2.1, they calibrate the sensors using Bouguet's stereo calibration implementation. As before, a global correction or optimization step is not described, so the calibration is done to the best of our knowledge in a pairwise manner.

Figure 2.1: Three Kinects setup used in Berger et al. [7].

2.1.2 Global Calibration and Refinements

For systems with three or more cameras, the approach by Svoboda et al. [67] is most frequently used, through the associated toolbox implementation [66]. It is an automatic method for calibrating both intrinsic parameters as well as the transformation between cameras simultaneously, by solving a linear system with respect to the camera parameters and the unknown 3D point locations. Outliers are pairwise removed via RANSAC and occluded points in the images are compensated for. The final parameters are refined using Bundle Adjustment and estimation of any non-linear distortion effects. The method is very robust but it cannot be used in setups with only two cameras. Berger et al. [8] evaluate the same approach for depth images in their multiple Kinect setup, however with inferior results compared to pairwise calibration.

A recent paper that also makes use of this method is by Tong et al. [69], where a novel approach for scanning humans with the use of depth sensors is presented. They employ three Kinects, positioned in close distance to each other so that each captures a partial view of a human positioned on a turntable. This configuration can be seen at Fig. 2.2. In reconstructing the final mesh, they register the partial views non-rigidly by explicitly using an approximation template, thus compensating for calibration errors. For this reason, their initial calibration is mostly generic; although not many details are mentioned, the approach by Svoboda et al. [66] is used in points captured by the RGB camera, followed by a pre-specified mapping from color to depth through OpenNI [4].

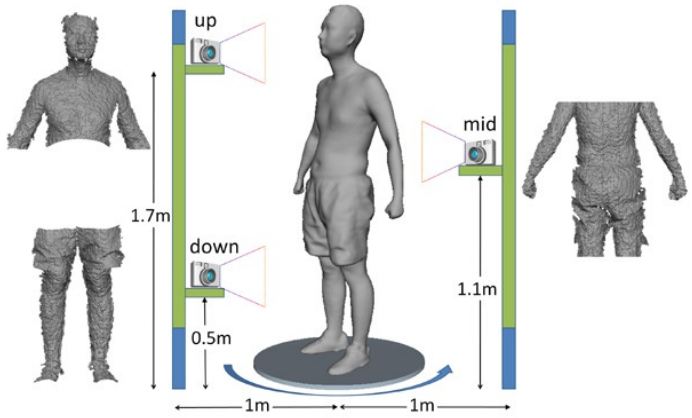


Figure 2.2: Three Kinects setup for scanning the human body used by Tong et al. [69].

Alexiadis et al. [5] use a setup of four Kinects positioned as shown in Fig. 2.3 for real-time capturing of moving objects. They globally calibrate the RGB cameras of the Kinects by combining the methods of Svoboda et al. [67] and Kurillo et al. [40]. They first use the algorithm proposed in [67] in order to detect points in the RGB color frames by waving a dual LED (red and green) in a dark environment. The calibration object can be seen in Fig. 2.3. The cameras are calibrated in pairs based on the epipolar geometry constraints (OpenCV implementation). A global optimization through Bundle Adjustment [49] is further applied for the estimation of extrinsic parameters. The total reprojection error after this second step drops significantly from 4 pixels to 0.84. This error however corresponds to the RGB and not the IR camera, so it is not unreasonable to assume that an additional, non-accounted for, error is introduced with performing the mapping from color to depth.

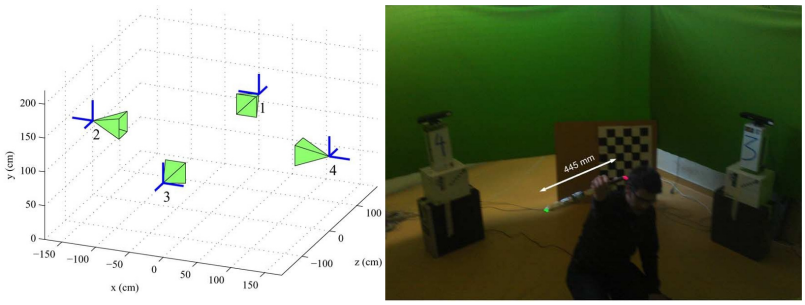


Figure 2.3: Four Kinects setup and calibration object used by Alexiadis et al. [5].

2.1.3 Other Methods

Susanto et al. [65] make use of a four Kinect system in order to combine the depth information for more robust object detection. The cameras are mounted on the ceiling and face the objects from different viewpoints. Due to the peculiarity of the setup, it is difficult to capture common points from all views and thus robust external calibration based on correspondences is difficult to achieve. For this reason, a manual calibration approach is performed instead of an automatic one; a camera is picked as a reference and the captured points cloud from the rest are manually transformed to the reference's coordinate frame. Their error however can go up to 13 cm for some views due to the noisy depth data and the position of the sensors.

In a paper by Microsoft Research with interesting applications, Wilson et al. [76] combine depth sensors and projectors, in order to create and explore different designs for interactive surfaces in a room. Two types of calibration processes are involved; cross-sensor calibration between cameras and projectors, which is not related to this thesis, and external calibration between the depth sensors. For the latter, a set of calibration points with known real-world positions is gathered, and the camera's

pose is estimated by applying the closed-form solution of Horn [34]. By applying this procedure for each camera separately, the individual positions with respect to the world coordinate system are computed.

2.1.4 Limitations of Existing Approaches

The aforementioned methods have a number of drawbacks and limitations which we aim to overcome in this work. Primarily, most of these methods do not differentiate between the conventional camera calibration and the depth sensor, i.e. the cameras are calibrated using points and methods for color-cameras. Even in cases where the points in depth image are used, only the locations on the depth map are taken into account and not the actual 3D coordinates. This does not account for errors due to misalignment between the different sensors (RGB, IR, depth).

Due to the depth error model of the structured light sensor, such generic approaches are not sufficient for external calibration. Note that in most cases, the setup is controlled in the sense that the "active volume" does not extend at a certain distance from the Kinect(s). For example, the object in [5] stands approximately 1.5 m away from the sensor, while in the scanning system of [69] the human is 1 m away. However, as we discuss in Chapter 3, the depth resolution drops quadratically with the distance increase from the device. This means that such calibration results will heavily depend on the distance of our scene from the sensors, with the error increasing as the depth data resolution and quality decreases. As a result, the requirements for a controlled set-up are strict.

Furthermore, in many scanning approaches any resulting approximation errors are compensated through a post-processing, global refinement step during the registration of the captured meshes, by relying on algorithms such as Iterative Closest Point (ICP). However, unless a template model, i.e. a prototype 3D shape, is used, the meshes need to have a fair amount of overlap in order for the refinement methods to work. The former limits the type and complexity of the acquired objects, i.e. template models are not always readily available or convenient to obtain and design. The latter, can restrict the physical sensor configurations, i.e. the distance or viewing angle, and produces an additional overhead in the setup design.

In our method, we aim to compensate for these types of errors of the sensor during the initial, and single, external calibration step, thus resulting in simpler, more accurate and more flexible configurations for capturing depth with multiple Kinects.

2.2 Capturing Calibration Points

Depending on the applied calibration method, calibration points can be captured on the color, infrared or depth frames (images). The most common method used in Computer Vision applications for conventional cameras is by utilizing a black and white (checkerboard) pattern, captured from multiple views. The checkerboard facilitates point detection, i.e. its corners are easily detectable using interest point detection methods such as the Harris corner detector. The resulting corners in the

different views and angles are used as the location of known 3D points in the acquired 2D images. However, even though this approach can be applied as-is in the color and infrared Kinect images, it cannot work in the depth since the corners are indistinguishable due to their coplanarity. A possible extension could be to adapt the method by choosing easily detected range image features, though the localization accuracy would be less than that with the image color features.

Berger et al. [8] propose a new way to extend the checkerboard method for the depth stream data. By replacing the squares with mirroring and diffuse surfaces, they manage to make the corners detectable by distinguishing between valid (diffuse) and non-valid (mirroring) pixel values, i.e. using a trivial, value-based reflection feature. The only assumption made is that the board is not orthogonal to the Kinect's viewing axis. This approach is shown in Fig. 2.4, where the difference between the conventional and the modified checkerboard can be seen.

Another method, again proposed in [8], for getting points from depth frames includes modifying a light source by attaching it at the center of a reflective disk (Fig. 2.4, right). The disk can be easily segmented due to invalid depth values and the center can be easily calculated. Instead of just picking the disk center though, an attached light is used as it is also detectable in the RGB frame. This is useful because, as will be shown in Chapter 3, it can be used for registering and calibrating the position of the Kinect IR and RGB cameras. This method however was reported to introduce a larger error compared to the calibration board [8].

Wilson et al. [76] use retro-reflective dots that can be distinguished in the IR frame and map their positions in the corresponding depth frame. In order to reduce

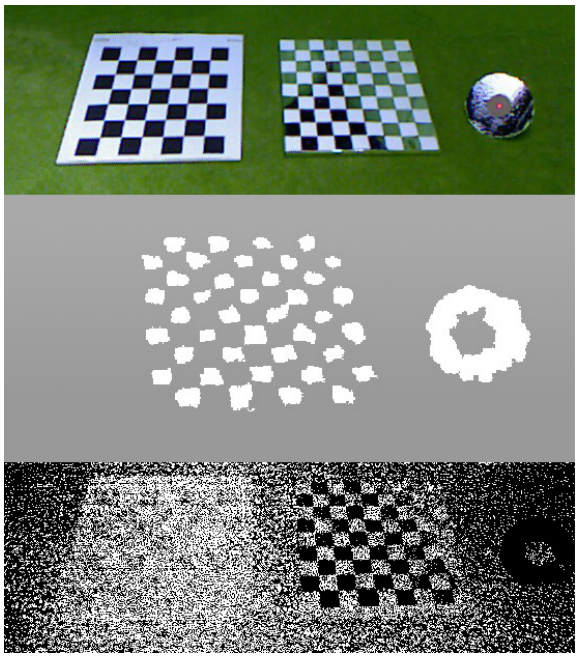


Figure 2.4: Calibration objects in RGB (top), depth (middle) and IR (bottom) frames used in [8]. The conventional calibration board (left) which cannot be detected on the depth frame, as opposed to the modified checkerboard with specular squares (center). The third object is a point light attached to a reflective disk (right), which can be segmented from the depth image and the light detected as the disk's center.

noise, the depth values are locally averaged for calculating a smoother depth value per position. Interestingly, note that this is the only case where actual 3D points are used, since in the aforementioned methods, only the point image coordinates (2D) are used during the calibration process.

In Chapter 4 we explore two approaches for obtaining calibration points; the first combines detection of board corners with mapping on the depth frame, and the second localizes points of interest directly in depth by using a spherical object.

Chapter 3

The Microsoft Kinect

3.1 Sensor Description

The Kinect sensor (Fig. 3.1) was developed by the Israeli company PrimeSense and released by Microsoft on November 4, 2010. Although it was initially marketed as a novel gaming peripheral for Microsoft's console, XBOX 360, its depth sensing capabilities and low-cost made it attractive for a variety of additional applications. It is now used in fields like Robotics, Human-Computer Interaction, Healthcare, Graphics, and Entertainment among others. In this section, we describe the sensor's components in more detail, explain its intrinsic parameters calibration procedure and introduce the depth model.

3.1.1 Kinect for XBOX vs Kinect for Windows

After the increase in the device popularity, Microsoft released the official Microsoft Kinect SDK on February 1, 2012 as well as a new sensor with slightly modified hardware and firmware targeted specifically for application development in Windows. There are minor differences between the two versions; Kinect for Windows has a near mode, enabling it to capture depth values as close as 40 mm (Fig. 3.2) as well as some enhanced SDK features such as better skeletal tracking. The main difference however, is that Kinect for Windows has a commercial license, thus enabling its use for commercial applications, something which is not possible with Kinect for XBOX.

Figure 3.1: The Microsoft Kinect device[1].

3.1.2 Hardware Components

Kinect's depth sensing system is comprised of an infrared (IR) emitter which projects a speckled pattern and an infrared camera that captures it. Apart from the depth sensing system, the device includes a number of additional components on the hardware level [54], which are visualized in Fig. 3.3:

[1] http://msdn.microsoft.com/en-us/library/hh438998.aspx
[2] http://msdn.microsoft.com/en-us/library/hh973078.aspx#Depth_Ranges

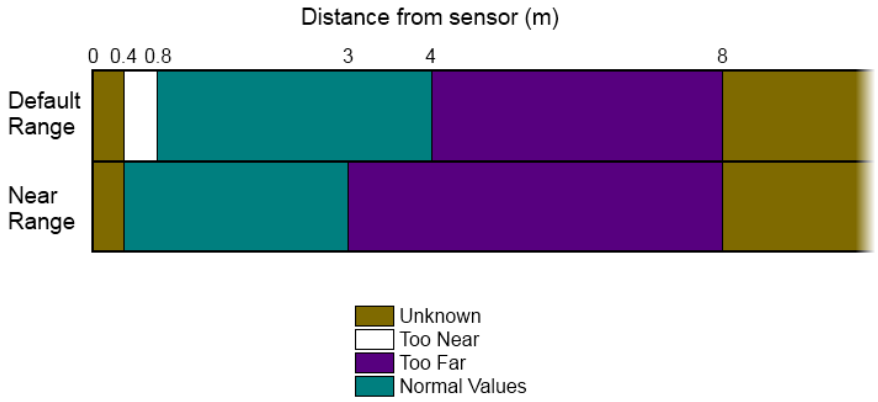


Figure 3.2: Valid depth values[2].

Infrared projector Emits an 830 nm infrared laser which gets diffracted through glass in order to create the projected speckled pattern.

Infrared camera A monochromatic CMOS sensor with resolution of up to 1280×960 and a field of view (FOV) of (57° Horizontal 43° Vertical) for recording the pattern formed by the IR projector.

Color camera A VGA color camera with a resolution of 640×480 at a frame rate of approximately 30 FPS. Resolution can go up to 1280×960, albeit with a reduction in frame rate (around 10 FPS). The returned color format can be RGB, YUV or Bayer [53].

Microphone array A four-microphone array for high quality audio capture with noise suppression and echo cancellation.

Tilt motor Controlled programmatically, with a pitch of $-27°$ to $+27°$.

Accelerometer 3-axis, for retrieval of orientation in relation to gravity.

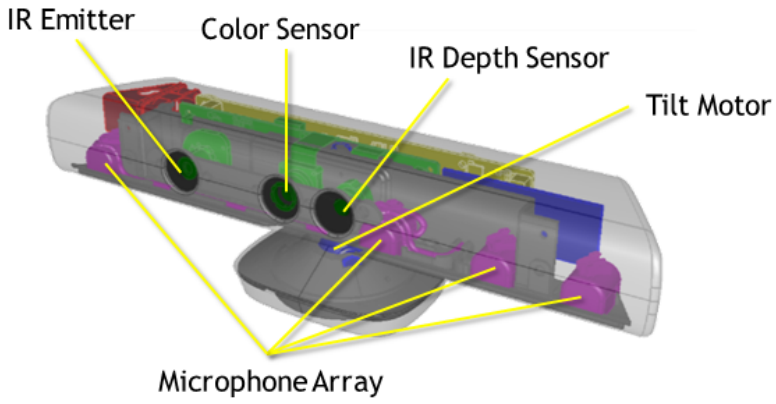


Figure 3.3: Microsoft Kinect components [54].

3.1.3 Software Drivers

Currently, there exist three different drivers for the Kinect. In this section, we briefly describe them and present a high-level comparison in Table 3.1. Note that only basic differences are noted, due to the continuous version updates for all three. The table should serve only as an indicator of which driver is suitable depending on the project requirements and specifications. A more thorough comparison can be found in [6] and [23].

OpenKinect [3] The OpenKinect community maintains the `libfreenect` library which was the first driver reverse-engineered shortly after the Kinect was introduced. It is the most basic of all three; it provides basic access to raw data and is not combined with high-level algorithms.

OpenNI [4] It is a joint effort of organizations, including PrimeSense, to establish standards for Natural Interaction (NI) devices. PrimeSense's NiTE middleware provides the Computer Vision algorithms, such as gesture recognition etc.

Microsoft Kinect SDK [1] The official SDK, has now reached version 1.7. Contains many high level features such as skeletal and face tracking and the latest addition, Kinect Fusion (real-time, high-quality reconstruction).

For our application we used both the Kinect SDK and OpenNI since not all of the required functionality was provided by one. For example, in the first version of the Windows Kinect SDK, no access to the IR image was provided, however this has changed with a recent version. Although some of the original development for this thesis was done with older drivers, all parts were eventually rewritten and adapted to the Windows Kinect SDK drivers. If not otherwise stated, the examples and results in the rest of this thesis will refer to this driver.

	OpenKinect	OpenNI	MS Kinect SDK
Cross-platform	Yes	Yes	No
Open-source	Yes	Yes	No
Crops depth	No	No	Yes at ∼4 m
Low-level access *(e.g. disparity image)*	Yes	No	No
High-level functions *(e.g. skeletal tracking)*	No	Yes through NiTE	Yes

Table 3.1: Comparison of available Kinect drivers.

3.2 Sensor Calibration

In order to get as accurate information as possible in terms of different data registration and localization on the scene, careful calibration is required for the two cameras on the Kinect, i.e. the IR and RGB, in the sense of determining both the intrinsic camera parameters as well as the relative position between them. The intrinsics of each camera involve nine parameters: two for the focal length, two for the center of projection, and five distortion coefficients. The relative extrinsics involve twelve parameters: nine for rotation and three for translation. Both the IR and the RGB cameras are factory calibrated and their parameters are burned to the firmware. However, even though the drivers provide us with functions to map the corresponding depth frame to color frame and vice versa, they do not explicitly supply neither the intrinsic values nor their relative position. As a result, the camera sensors need to be explicitly calibrated prior to any subsequent steps.

3.2.1 Color Camera Intrinsics

The color camera is a conventional RGB camera and is calibrated as such. The most common approach for color camera calibration is based on the method by Zhang [79, 80] and the intrinsics model in [30]. This approach is the one implemented in the frequently used Bouguet MATLAB Camera Calibration Toolbox [36] and in the OpenCV library [2]. A checkerboard calibration pattern is used from many frames in various poses. The pattern corners are detected using Harris corner detection and optimized to subpixel accuracy with gradient-search. Using these points and

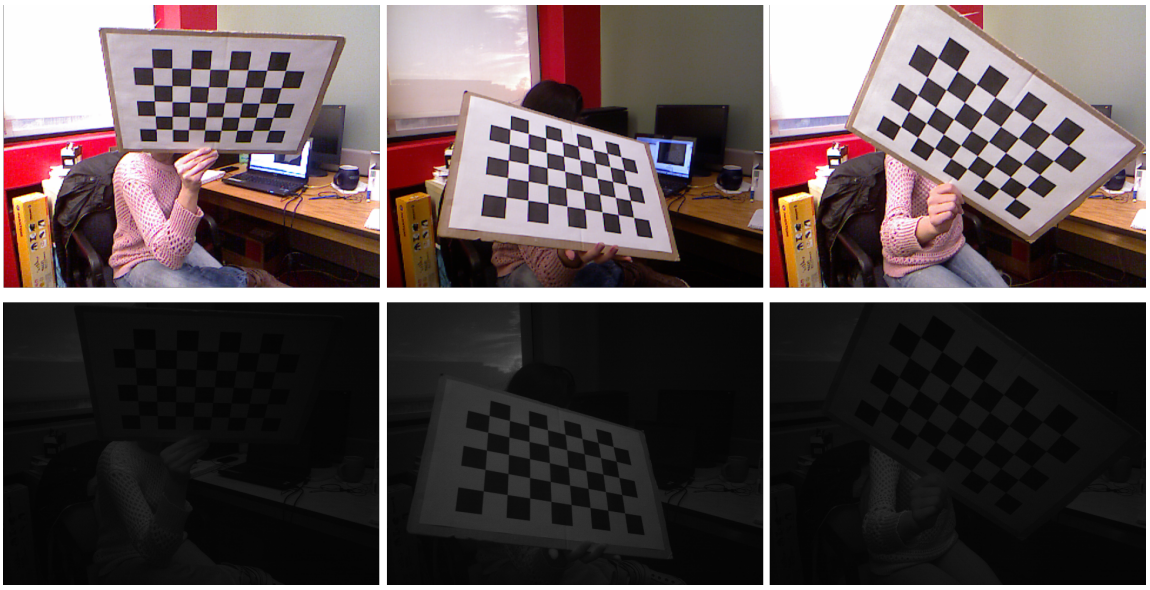


Figure 3.4: RGB and corresponding IR frames used for intrinsics calibration.

the corresponding 3D, from the known geometry of the planar object, an initial least squares solution is computed. The objective function to be minimized is the reprojection error, i.e. the average difference of the true 2D to the projection of 3D points using the estimated projection matrix. Optimization is done using the Levenberg-Marquardt algorithm [49], and in order for the calibration to be deemed successful, the reprojection error must be less than one pixel. In Tables 3.2 and 3.3, we present some representative values, estimated for one of the used devices in this thesis. It should be noted that since each Kinect device has similar, but subtly different values, these are provided as an indication of range and order of magnitude and should not be used by default for any Kinect device.

parameter	value
focal length	(511.772, 513.360)
center of projection	(322.459, 255.144)

Table 3.2: Intrinsic parameters for RGB camera.

distortion coefficient	value
k1	0.2242
k2	-0.7976
k3	0.9466
p1	0.0102
p2	0.0014

Table 3.3: Distortion coefficients for RGB camera.

3.2.2 Infrared Camera Intrinsics

The infrared camera calibration is very important because the coordinates of the point cloud that will be created from the depth map depend on the camera's intrinsic parameters, as will be analyzed in Chapter 5. The calibration is done in a similar fashion to the RGB camera, with a small modification in the way the frames are captured. The dot pattern that is emitted from the IR projector makes it more difficult for the corner detection algorithm to locate the chessboard corners. For this reason, before capturing the frames, the projector is explicitly covered to mask the emission. Some representative images taken with the IR camera can be seen in Fig. 3.4. Tables 3.4 and 3.5 show indicative intrinsic and distortion parameters

respectively, for one of the used devices.

parameter	**value**
focal length	(572.153, 573.784)
center of projection	(317.434, 248.140)

Table 3.4: Intrinsic parameters for IR camera.

distortion coefficient	**value**
k1	-0.0985
k2	0.3655
k3	-0.5360
p1	0.0065
p2	-0.0003

Table 3.5: Distortion coefficients for IR camera.

3.2.3 Color-IR Stereo Calibration

The baseline between the two cameras of the Kinect is approximately 2.5 cm [60]. This means that the captured color frames do not align exactly and thus do not directly map to the depth frames (Fig. 3.5). In order to establish an accurate color-to-depth mapping, extrinsics calibration is performed in order to estimate the relative pose (rotation and translation) between the two camera coordinate systems.

A straightforward extension of single camera calibration, is to calibrate each

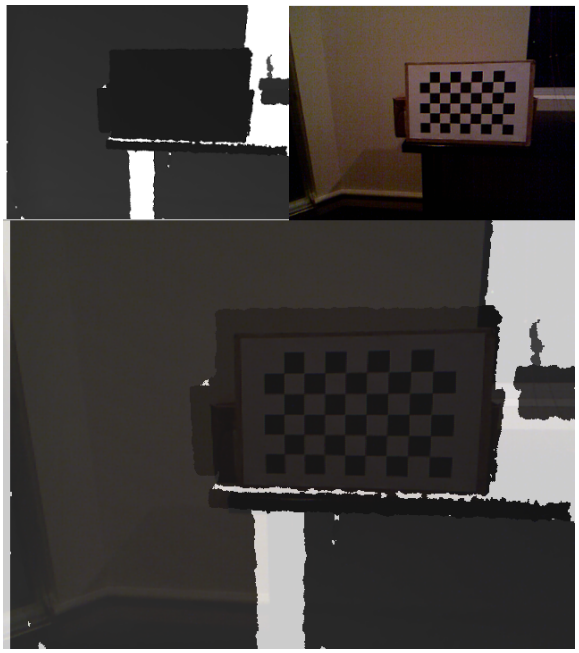


Figure 3.5: Uncalibrated color frame mapped to corresponding depth.

camera independently, with respect to their extrinsics and given their intrinsics, and estimate or calibrate for the relative pose [30, 80, 78]. Herrera et al. [33] present a method specifically for color and depth sensors that is based on simultaneous color and depth features to globally refine the calibration of the two cameras. The method requires only a checkerboard pattern and a planar surface with manual corner specification.

Since color mapping is not crucial in our application and we can tolerate minor errors, we chose the conventional stereo calibration (Chapter 2), adapted for the case of a color-IR pair [80, 78]. The method is described below in its more general, camera-pair calibration from multiple views. As a side note, it can also be used in order to calibrate a Kinect sensor with an external RGB camera.

Generic Multi-view Stereo

The main idea is that the fixed, relative position of the two cameras $[R \,|\, T]$ can be estimated from the individual poses of an object with respect to the first and second at positions $[R_1 \,|\, T_1]$ and $[R_2 \,|\, T_2]$ respectively. By R and T we denote the 4×3 and 4×1 augmented rotation and translation matrices of rotation (9) and translation (3) parameters of the implied transformation. The solution is provided by solving the system:

$$R_2 = R * R_1, \qquad T_2 = R * T_1 + T. \tag{3.1}$$

This is achieved through a global optimization of the estimates for the individual extrinsics from multiple captures/frames in a least-squares sense:

$$[R \,|\, T] = \underset{R,T}{\operatorname{argmin}} \sum_{j=1}^{M} \left\| [R_{2j} \,|\, T_{2j}] - [R \,|\, T] * [R_{1j} \,|\, T_{1j}] \right\|_2^2, \tag{3.2}$$

with $E_{1j} = [R_{1j} \,|\, T_{1j}]$ the 4×4 parameter matrices for a single frame. These can be estimated based on the pinhole camera model and the estimation of a perspective projection using point correspondences between world-image frame. Assume a set of N world 3D point locations, in each out of M views $\{X_{ij}\}, i = 1 \ldots N, j = 1 \ldots M$ and the corresponding 2D points $\{x_{1ij}\}$ and $\{x_{2ij}\}$ in the images from the two cameras respectively. For each camera separately and each view, the total re-projection error is minimized using the global Levenberg-Marquardt algorithm:

$$E_{1j} = \underset{E_{1j}}{\operatorname{argmin}} \sum_{i=1}^{N} \left\| x_{1ij} - K_1 E_{1j} X_{ij} \right\|_2^2, \tag{3.3}$$

where K_1 the matrix of intrinsic parameters, specified in advance, as highlighted in the previous subsections.

Combining Eq. 3.2 and 3.3, the total minimization problem that is being solved for multi-view, stereo calibration can be written and formulated as:

$$\sum_{j=1}^{M}\sum_{i=1}^{N}\left(\left\|x_{1ij}-K_1[R_{1j}\,|\,T_{1j}]X_{ij}\right\|_2^2+\left\|x_{2ij}-K_2[R_{2j}\,|\,T_{2j}]X_{ij}\right\|_2^2\right)+ \\ \sum_{j=1}^{M}\left\|[R_{2j}\,|\,T_{2j}]-[R\,|\,T]*[R_{1j}\,|\,T_{1j}]\right\|_2^2. \quad (3.4)$$

The above, total re-projection error for all points, views and cameras, is minimized with respect to all extrinsics, for both cameras and all views $\{R_{1j}, T_{1j}, R_{2j}, T_{2j}\}, j = 1 \ldots M$ and the relative position or calibration transformations $\{R, T\}$ between the two.

Specific Color-IR Framework

The practical framework involves the following steps: We capture a checkerboard in various poses from both IR and color cameras and solve for the relative rotation and translation similar to Eq. 3.4 using the correspondences defined by the 3D board and detected image corner points. One drawback of this setup is that no drivers can stream both RGB and IR channels simultaneously. In order to surpass this, we have designed an interface that changes modes automatically very fast; the user presses a button to capture the frame, and, after the IR image is saved, the mode changes and captures the corresponding RGB. This implies that the board should be still for the milliseconds it takes to capture both frames, which can be guaranteed by using a fixed tripod or similar mounting.

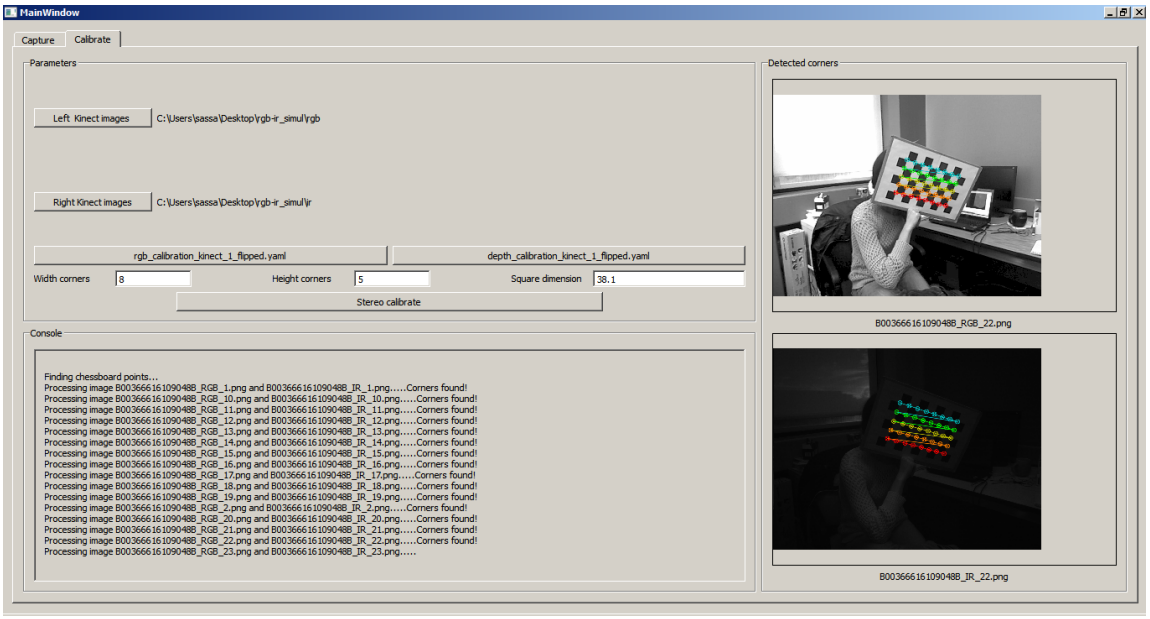


Figure 3.6: RGB-IR calibration interface.

3.3 The Depth Capturing System

In this section, we describe the process behind Kinect's depth value acquisition and illustrate the various sources of error in the sensor.

3.3.1 Acquiring Depth Data

Before we proceed, it is important to make a crucial distinction; the Kinect does not capture the actual distance of the object from the camera, but rather the distance of the plane of the object to the camera, i.e., the projection of the actual distance on the camera viewpoint. This point is illustrated in Fig. 3.7. In order to infer depth for each position, the Kinect uses the structured light principle. It finds a disparity

map by using the correlation between two light patterns: a stored reference one and the one observed from the scene. This process is described in more detail in the following sections.

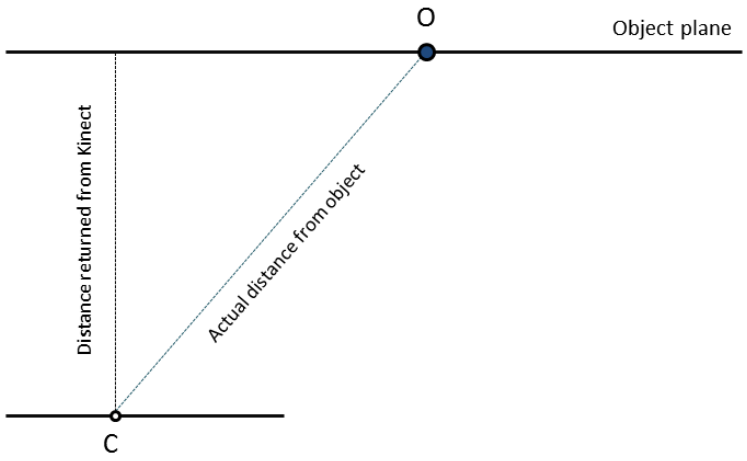


Figure 3.7: Actual depth distance measured by the Kinect sensor.

Getting the Disparity Map

The IR projector emits a laser beam, which with the use of a diffraction grating creates a pseudo-random pattern of speckles that gets projected onto the scene (Fig. 3.8). This pattern is then captured by the IR camera with the use of a band-pass filter centered at the infrared laser wavelength, and compared to a reference pattern. This reference pattern is burned to the firmware and is basically the same laser grid projected in known measured depth. By comparing the shift of corresponding dots through a 9×9 correlation window per dot [60, 17], a disparity map is produced, which is then used to infer the depth at each pixel. The disparity representation uses 11 bits, thus providing 2048 quantization levels.

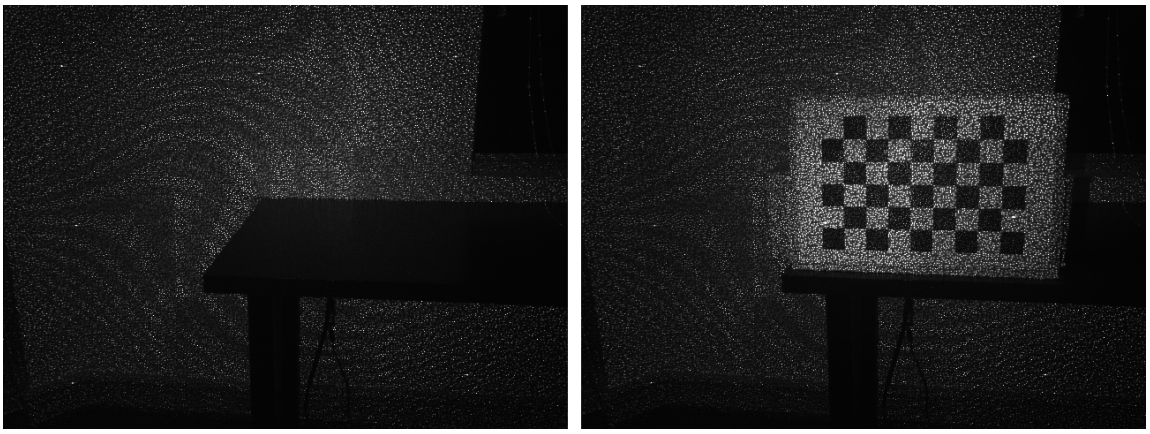


Figure 3.8: The IR speckled pattern emitted by the laser projector.

Getting the Depth Map

The depth map is acquired by the disparity value of each pixel through a triangulation process. The relationship between disparity and depth can be derived geometrically by looking at Fig. 3.9, where L symbolizes the infrared (laser) projector and C the infrared camera. We assume P_r is the position of a speckle in a known reference depth Z_r and P_o is the same dot captured by the Kinect at an object depth Z_o. D symbolizes the 3D disparity between them and d is the corresponding disparity on the 2D image plane that the sensor computes. The parameters f and b symbolize the focal length and baseline, i.e., the distance between IR projector and IR camera, respectively. From the similarity of triangles in Fig. 3.9 we get:

$$\frac{D}{b} = \frac{Z_r - Z_o}{Z_r}, \tag{3.5}$$

$$\frac{d}{f} = \frac{D}{Z_o}. \tag{3.6}$$

Thus, by combining Eq. 3.5 and Eq. 3.6, we get the depth value Z_o:

$$Z_o = \frac{Z_r}{1 + \frac{Z_r}{fb}d}. \tag{3.7}$$

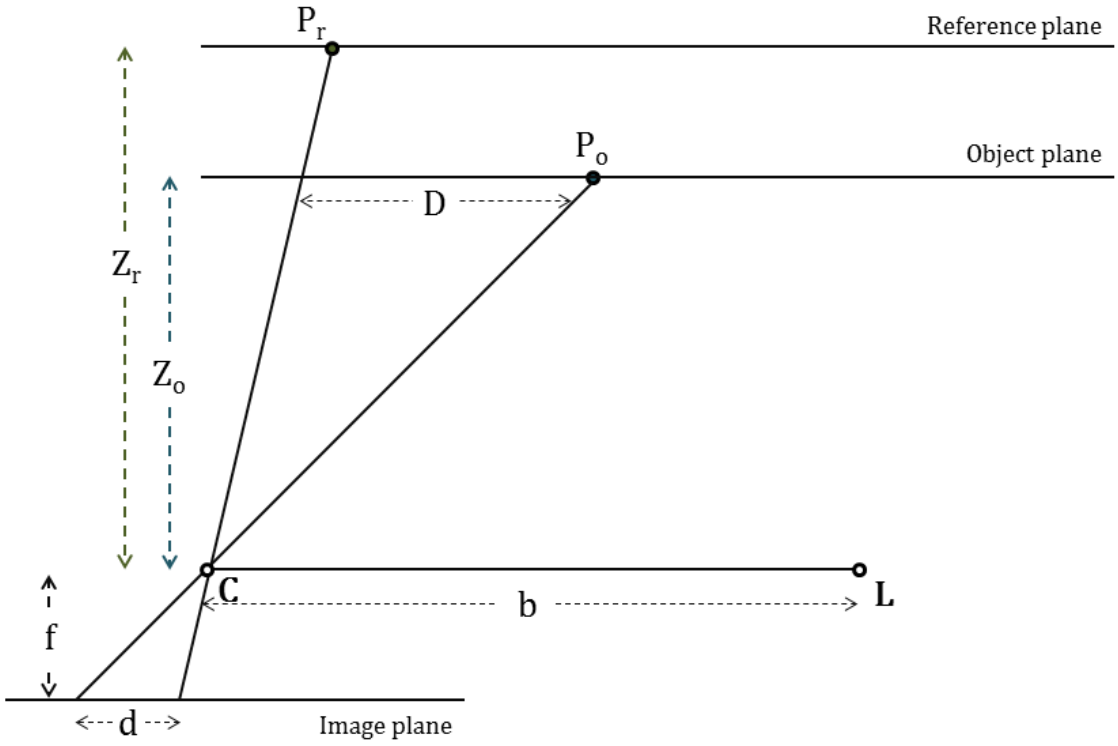


Figure 3.9: The triangulation process for depth from disparity.

3.3.2 The Depth Error Model

Correcting any errors due to the depth acquisition process after calibration, similar to undistorting the images from a conventional color camera, is not as straightforward. In effect, merely undistorting the image captured from the IR camera is not enough, and may additionally introduce further distortions due to the correlation-based disparity estimation. The depth sensor error model is more complex, and

includes additional sources of error, due to both the sensor and the set-up, which we enumerate in this Section.

Errors Due to the Sensor

With respect to the sensor, there are three distinct sources of error which correspond to the different parts of the depth calculation process. Overall, errors are introduced from the intrinsic distortion of the infrared camera that captures the scene, the transformation of disparity to depth values, and the inherent quantization of the disparity measurement.

I. Errors Due to the IR camera

The infrared camera does not conform to the ideal pinhole camera model, and thus its lens introduces radial and tangential distortion. These parameters are estimated through the intrinsic calibration of the IR camera, which was previously described. This process however involves solving for a most-likely overdetermined linear system and results in one solution by a very close approximation. Nevertheless, in the Kinect case these distortion coefficients are negligible. This agrees with our observations on obtaining worse depth mapping results when applying undistortion, with the very small estimated coefficients. Note though, that this may potential be attributed to frames being already returned undistorted from the Kinect SDK.

II. Errors During Triangulation

While converting disparity values to depth using triangulation, we rely on two

extra parameters intrinsic to the sensor: the baseline between the projector and the IR camera, and the depth of the reference laser grid, burned to the device. However, these values need calibration only when working with the OpenKinect drivers, since these return the raw disparity frame and the user is responsible for converting it to depth. For example, Stone et al. [64], after normal calibration of the infrared camera, use a large number of training points to optimize for the baseline, focal length, and distortion parameters. This is not necessary though for the Microsoft SDK and the OpenNI drivers, since these explicitly make use of the factory calibrated values.

III. Errors Due to Depth Resolution

The accuracy of the depth map depends significantly on the distance of the scene from the sensor. Smisek et al. [63] measured experimentally the resolution of the device by capturing planar targets from consecutive depths, ranging from 0.5 m to 15 m, and expressed the distance between two recorded values, referred to as the *quantization step* q, as a function of the depth z:

$$q(z) = 2.73z^2 + 0.74z - 0.58 \ \text{[mm]},$$

with z in meters. We borrow their example to demonstrate how large q can get: at a distance of 0.5 m we have $q(0.50) = 0.65$ mm, but for a depth of 15.7 m, $q(15.7) = 685$ mm, i.e. *a* 10^2 *increase in order of magnitude*. Besides this decrease in depth resolution, Khoshelham [38] also reports a reduction in point density as we move away from the sensor. This error is inherent to the sensing device and cannot be eliminated. Since it is not negligible, even for close distances, there is a need for methods that compensate for it implicitly.

Errors Due to the Setup

An additional type of errors due to the capturing configuration is easily controllable by specifying acquisition protocols. The most important one concerns the surface material of the object captured. The Kinect does not respond well to shiny materials, and specular surfaces result in invalid values. Furthermore, depending on the scene configuration, we can get invalid values and occurrences of shadowing for the following reasons: Either a part of the scene is illuminated by the projector, but it cannot be seen from the camera, or it is imaged by the camera, but there is no laser illuminating it.

Chapter 4

Non-rigid Calibration

An essential step in a setup involving multiple cameras is the calibration of their external positions. By calibration, we refer to estimating a transformation that can be applied to a camera's coordinate system in order to be transformed to a reference one. The reference coordinate system refers to a single camera system, e.g., a frontal-looking one, that serves as the one with respect to which all other relative positions or transformations will be estimated.

Problem statement: Let's assume that we have captured the same N calibration points from two different viewpoints using Kinect cameras K_1 and K_2. Each Kinect returns a depth map according to the distance of the points from its own coordinate system. Consequently, point values returned from K_1 will refer to a coordinate system origin $(0, 0, 0)$ at the location of K_1, while those returned from K_2 will assume a different origin at the location of K_2. Therefore a mapping to a common coordinate system is required in order to form the composite set of points from both

cameras.

As we shall see in the next Chapter, we can use the depth value z returned from the Kinect to retrieve the values of x and y, thus obtaining for every calibration point the triplet (x, y, z) of real world coordinates. To reflect this, we will use the following notation for the calibration points: We will symbolize the point coordinate values returned from K_1 for a single point as the homogeneous vector $p_i = [x_{p_i}, y_{p_i}, z_{p_i}, 1]^T$ and the set of N calibration points as $\{p_i = [x_{p_i}, y_{p_i}, z_{p_i}, 1]^T,\ i = 1 \dots N\}$. Correspondingly, the set of points returned by K_2 will be denoted by $\{q_i = [x_{q_i}, y_{q_i}, z_{q_i}, 1]^T,\ i = 1 \dots N\}$. In matrix notation, the corresponding $4 \times N$ matrices of column-wise concatenated point coordinates will be $P = [p_1 \dots p_N]$ and $Q = [q_1 \dots q_N]$. To avoid any confusion, we will keep this notation of points p_i and q_i for K_1 and K_2 respectively, consistent throughout this chapter.

As a convention, we choose K_2 as the reference coordinate system (Fig. 4.1), thus the problem now becomes the estimation of the transformation M that will map point coordinates from K_1 to K_2.

In this chapter we show that a rigid model is not enough to describe the transformation between two depth sensors, due to the non-linear nature of errors (and noise) that appear in the observed scene. Towards a more accurate calibration process, we first find the underlying affine transformation with the help of automatic calibration points and then extend it to a non-affine component, after discussing and supporting the inadequacy of the rigid assumption.

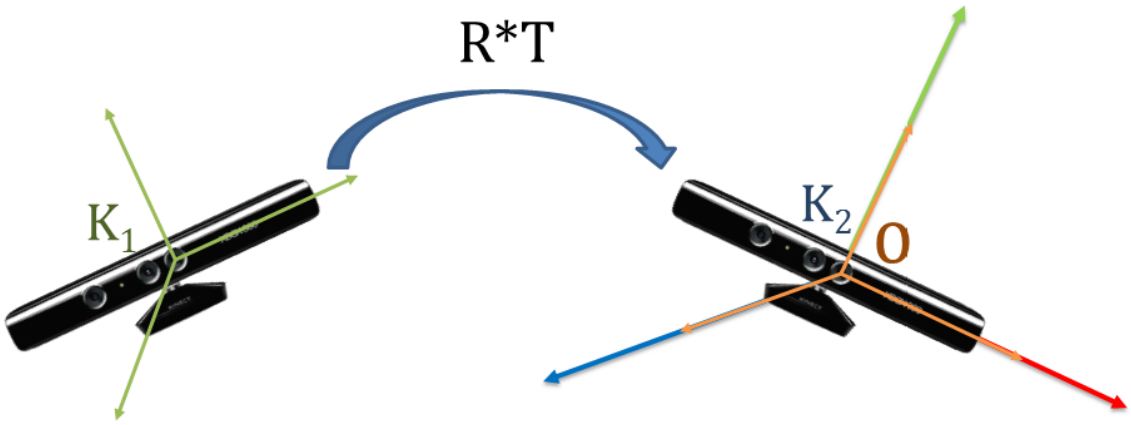


Figure 4.1: Two local coordinate systems and a global one, coinciding with the location of one of two Kinect sensors.

4.1 Initial Stereo Calibration

The Euclidean transformation M that maps a coordinate system to another is composed of a rotation R and translation T and can be found from the calibration point clouds' correspondences. Ideally, transforming the matrix of points P using M, means that it will coincide with the matrix of points Q viewed from the second camera. We therefore get the linear mapping:

$$Q = MP, \quad P = [p_1 \dots p_N], \quad Q = [q_1 \dots q_N], \tag{4.1}$$

where the linear transformation M can be written with respect to the individual rotation R and translation T parameters:

$$M = RT = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & t_x \\ 0 & 0 & 0 & t_y \\ 0 & 0 & 0 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Eq. 4.2 shows the decomposition of M into the rotation and translation matrices R and T, expressed in homogeneous coordinates. In total nine rotation and three translation parameters describe the mapping and require four point-correspondences to be completely specified.

From Eq. 4.1 and Eq. 4.2 we get the following linear system:

$$\begin{bmatrix} x'_1 & \cdots & x_N \\ y'_1 & \cdots & y_N \\ z'_1 & \cdots & z_N \\ 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_N \\ y_1 & \cdots & y_N \\ z_1 & \cdots & z_N \\ 1 & \cdots & 1 \end{bmatrix}, \tag{4.3}$$

where the two matrices R, T have been combined into one 4×4 homogeneous transformation matrix with the corresponding translation parameters being:

$$T_x = r_{11}t_x + r_{12}t_y + r_{13}t_z$$

$$T_y = r_{21}t_x + r_{22}t_y + r_{23}t_z$$

$$T_z = r_{31}t_x + r_{32}t_y + r_{33}t_z$$

The above linear system, with 12 unknown parameters, is overdetermined in the case of $3N > 12$ equations, i.e. $N > 4$ point correspondences between the two clouds. Since an exact solution is not feasible for an arbitrary number of points N, we will approximate M using the Least Squares formulation [46]; we are searching for M which minimizes the norm of point transformation error $\|Q - MP\|_F^2$. More formally:

$$M = \underset{M}{\operatorname{argmin}} \left\{ \|Q - MP\|_F^2 = \sum_{i=1}^{N} \|q_i - Mp_i\|^2 \right\}. \tag{4.4}$$

A solution to the Least Squares problem can be obtained using factorization through Singular Value Decomposition (SVD) [27]. SVD is a matrix factorization method, according to which, an $m \times n$ matrix A can be written in the form:

$$A = USV^T,$$

where U is an $m \times m$ orthogonal matrix ($UU^T = U^TU = I$), whose columns are the orthonormal eigenvectors of AA^T; V is an $n \times n$ orthogonal matrix whose columns are the orthonormal eigenvectors of A^TA and V^T is its transpose; S is a diagonal $m \times n$ matrix that contains the square roots of the non-zero eigenvalues of either U or V (their non-zero eigenvalues are in fact the same, so it makes no difference).

By using SVD, we can obtain the pseudo-inverse of P, symbolized with $P^\dagger$, which can be used in order to solve Eq. 4.1 with respect to M as follows:

$$M = QP^\dagger = QVS^\dagger U^T. \tag{4.5}$$

Here, U and V follow their aforementioned definition, while $\mathrm{S}^\dagger$ is the pseudo-inverse of S which is obtained by taking its transpose, after replacing every non-zero diagonal value (s_{ii}) with its reciprocal ($1/s_{ii}$).

4.2 Insufficiency of the Rigid Assumption

In Chapter 3 we analyzed in detail the sources of error that can affect the accuracy of the measurement results. In this section, we link these errors to the overall calibration process in an effort to understand how they render the rigid calibration inadequate for the case of the specific depth sensors.

In Fig.4.2 we illustrate how the capturing of one point from both cameras is affected by errors and the displacement changes these errors introduce. The point

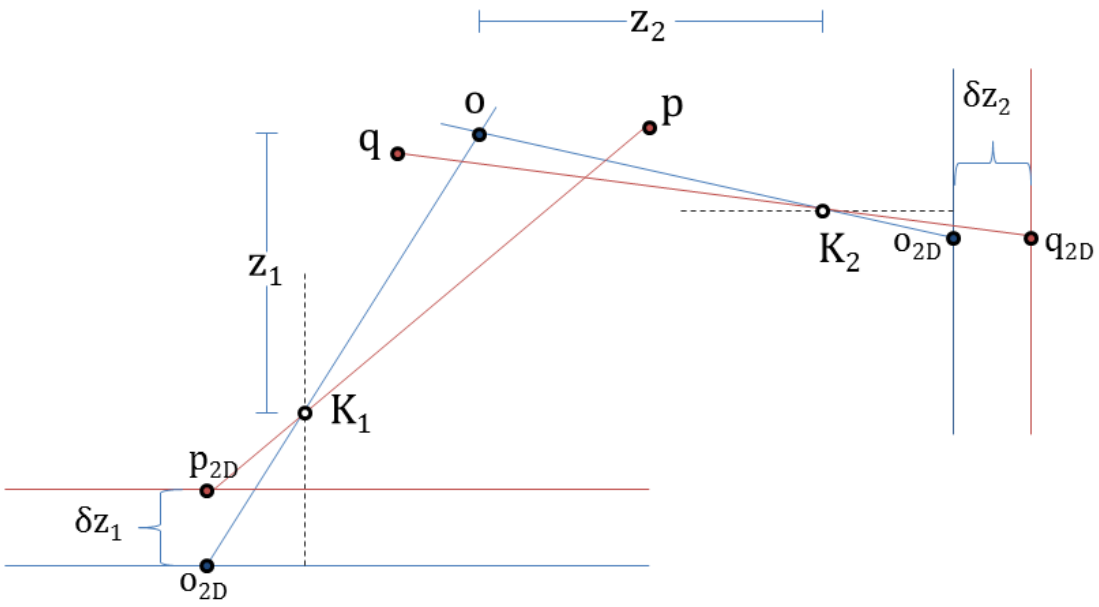


Figure 4.2: Error during depth capturing.

o is captured simultaneously by K_1 and K_2 which are located at depth distances z_1 and z_2 respectively from it. The errors introduced by the cameras here are cumulative and originate from three sources: calibration parameters of the IR sensor, calibration parameters during triangulation and discretization (quantization) errors

due to the varying Kinect depth resolution. The first two are relatively small and do not contribute much to the overall error. They can also be potentially corrected or mitigated by carefully designing the calibration process. However, the depth resolution errors δz grow quadratically with depth (Section 3.3.2), are inherent in the acquisition process and cannot be eliminated.

As depicted schematically in Fig.4.2, K_1 incorrectly captures the depth of point o at the depth corresponding to the position of point p, i.e. a depth difference of δz_1 units. Likewise, K_2 acquires point o with depth corresponding to q. Note that the difference between the intended o and the position of the backprojected point p from the measurement, is in all 3D dimensions and not only along the z axis. This happens because depth z is explicitly incorporated in retrieving the x and y coordinates and translates in a 3D backprojection error.

In Fig.4.3 the error difference is shown for the case two calibration points. Since the transformations between the two cameras will be specified by a separate displacement e_i for each pair of points, we cannot use the same transformation in order to transform points P to points Q. The parameters of the transformation depend on the seen point distance e.g. for points close by it could be a translation of some units (centimeters), less than for points further away. This in turn translates to the actual point distance from each of the sensors. Note that, if the distances were the same, a rigid, affine transformation defined by four pairs of points would suffice to specify a mapping.

In Fig. 4.3 we show a set of calibration points captured simultaneously from two Kinects, K_1 (red) and K_2 (blue), transformed to the same coordinate system using

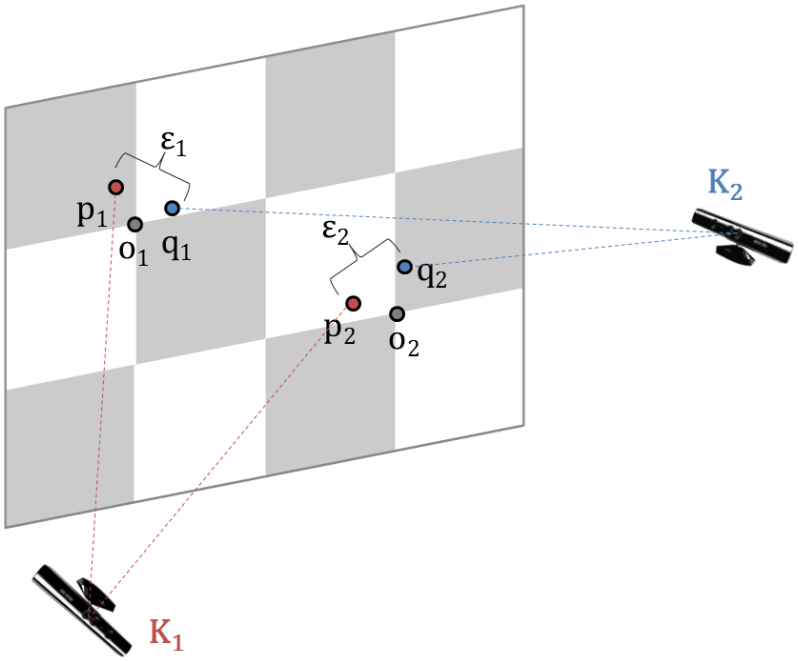


Figure 4.3: Error difference in calibration points captured by two sensors.

the affine matrix, estimated from the initial rigid calibration step. Ideally those points should overlap or be approximately overlapping, however, we can actually observe the error discussed in the previous Section; the corresponding points exhibit a shift which is different for each pair. This deviation during calibration translates to an error when registering a scene captured from the two cameras. Unless the two views of the scene contain a sufficiently overlapping region of points, *correcting for this error during the registration step* becomes very challenging, in the sense that existing algorithms and approaches cannot be applied. Thus, our goal is to model and *compensate for this type of error during the calibration step*. Motivated by the non-linearity that the depth error exhibits, we propose to use a non-rigid transformation model in order to capture these variations, possibly on top of an

initial affine transformation.

4.3 Non-rigid Correction

Formulation: According to our notation, we are using N calibration points and use Q to symbolize the matrix of points $q_i, i = 1 \dots N$. For our analysis, we consider Kinect K_2 as our reference system, thus we adopt the notation P' for the matrix P of points p_i, transformed to K_2 using an affine, Euclidean transform, estimated through an overdetermined linear system, as in Sec. 4.1. Our goal is to refine that estimate by finding a mapping that captures the non-rigid transformation that maps P' to Q. This will have to be a closed-form solution in order to apply it offline for aligning the point clouds of the captured scenes. In essence, such a solution will function as *calibration function* for the registration of any unseen acquisitions.

4.3.1 Thin Plate Splines

We will be using an implicit representation f to model the minimum deformation that should be applied to a surface passing through points P' in order to map exactly to desired positions Q. Note that the correspondences between points must be known in advance, which is inherent to our calibration setup. We formulate this as a Thin Plate Splines (TPS) mapping, due to the attractive properties of the TPS framework: It provides a closed form solution which is easy and fast to compute for

a small number of points, and reduces to a regular affine transformation when possible. The latter means that, due to the affine component, the deformation will be as smooth as possible, thus reducing the possibilities of overfitting. Thin Plate Splines were first mentioned as interpolation functions by Duchon [24] and were more extensively studied by Bookstein [11] and Wahba [73] for the description and modeling of deformations. Even though the original formulation was developed for 2D functions, it was generalized and effectively applied for interpolation and deformation mapping in 3D [71, 15, 59, 56]. Intuitively, as the name suggests, the Thin Plate Spline corresponds to the deformation that would be applied to a thin metal sheet if it had to pass through specific points, called control points (Fig.4.4).

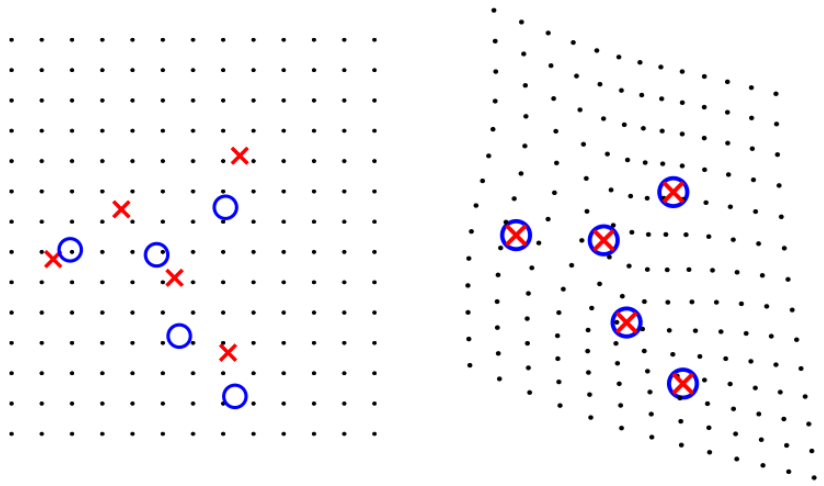


Figure 4.4: Thin Plate Splines interpolation (from [22]).

Mathematically, the mapping or deformation corresponds to the function f which minimizes the sum of two energy terms; the interpolation energy E_i and the bending

energy E_b. In more detail, the interpolation energy

$$E_i = \sum_{i=1}^{N} \|q_i - f(p_i')\|^2 \tag{4.6}$$

requires that the function passes through all control points by penalizing the mapping difference error. Minimizing the bending energy

$$\begin{aligned} E_b = \iiint \Bigg\{ &\left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 + \left(\frac{\partial^2 f}{\partial z^2}\right)^2 \\ &+2\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial z}\right)^2 + 2\left(\frac{\partial^2 f}{\partial y \partial z}\right)^2 \Bigg\} dxdydz \end{aligned} \tag{4.7}$$

enforces f to be as smooth as possible by penalizing the interpolating surface curvature. We are looking for a function f that minimizes the energy functional:

$$E_{tps} = E_i + E_b \tag{4.8}$$

In practice however, the energy function 4.8 becomes:

$$E_{tps} = E_i + \lambda E_b \tag{4.9}$$

where λ is a regularization parameter that controls the trade-off between smoothness and interpolation. If $\lambda = 0$ or very small, we have a scheme with exact interpolation and limited smoothness constraints, while as λ grows larger, f becomes very smooth but may not represent the data accurately. The regularizer is mostly used when there is noise in the data, or the number of control points is small. In our case, and since all collected points are used as control points, λ is set to the smallest non-zero value, unless otherwise noted.

The sought function is then the solution to the minimization problem:

$$f_{tps} = \underset{f}{\operatorname{argmin}}\{E_{tps}\}. \tag{4.10}$$

Notice how this is reminiscent to the classic formulation of supervised regression, where a function f is learned from a set of known point (q_i, p'_i) matches, using higher-order regularization constraints to avoid overfitting.

4.3.2 Approximating the Mapping Function with RBFs

A Radial Basis Function (RBF) is a function of the form $\phi(r)$, where $r = \|p - p_i\|$ usually represents the Euclidean distance between two points p and p_i. RBFs are widely used in various fields in order to build function approximations using finite weighted sums of the form:

$$f(p) = \sum_{i=1}^{N} w_i \phi(\|p - p_i\|) \tag{4.11}$$

where w_i is a weight associated with each of a set of reference or sample points $\{p_i\}$.

In our case, RBFs are used to approximate the non-rigid part of the transformation function f, which is of the form:

$$f(x, y, z) = \underbrace{a_1 + a_2 x + a_3 y + a_4 z}_{\text{affine}} + \underbrace{\sum_{i=1}^{N} w_i \phi(\|(x, y, z) - (x_i, y_i, z_i)\|)}_{\text{non-affine}} \tag{4.12}$$

The function includes also an affine transformation part $\mathbf{a}^T p = a_1 + a_2 x + a_3 y + a_4 z$, that accounts for the rigid part of the deformation. The affine $\mathbf{a} = [a_1, a_2, a_3, a_4]^T$ and non-affine, RBF weights $\mathbf{w} = [w_1, \ldots, w_N]^T$ are the unknown parameters of the mapping function.

Various types of RBFs have been explored, such as the (2D) thin plate spline $\phi(r) = r^2 \log(r)$, the Gaussian $\phi(r) = e^{-(r/\sigma)^2}$, or the multiquadric $\phi(r) = \sqrt{1 + r^2}$.

In the 3D space Thin Plate Splines formulation, we use those of the biharmonic spline, i.e. $\phi(r) = r$ and thus Eq. 4.12 becomes:

$$f(x, y, z) = a_1 + a_2 x + a_3 y + a_4 z + \sum_{i=1}^{N} w_i(\|(x, y, z) - (x_i, y_i, z_i)\|). \tag{4.13}$$

The biharmonic spline was shown to be the smoothest interpolant in R^3 [24] and it is suitable for non-uniformly sampled data.

Finding the TPS Coefficients

Under the requirement of square-integrable second derivatives for f (Eq. 4.7), we require the following constraints:

$$\sum_{i=1}^{N} w_i = \sum_{i=1}^{N} w_i x_i = \sum_{i=1}^{N} w_i y_i = \sum_{i=1}^{N} w_i z_i = 0. \tag{4.14}$$

We also have the following interpolation conditions stemming from Eq. 4.6:

$$q_i = f(p_i), \qquad i = 1, \ldots, N. \tag{4.15}$$

The conditions from 4.14 and 4.15 can be written as a linear system of the equations for the unknown TPS coefficients $\mathbf{w}, \mathbf{a}$ as follows:

$$\begin{bmatrix} K & P' \\ P'^T & O \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} Q \\ o \end{bmatrix} \tag{4.16}$$

When the regularization parameter λ is included the submatrix K becomes $K + \lambda I$.

To make the structure of the system more explicit, we provide the analytic form

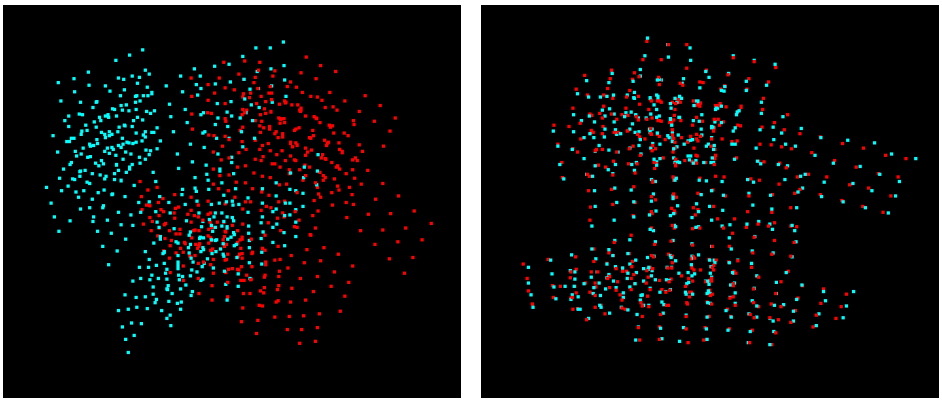
in our case. With the left-hand side $(N+4) \times (N+4)$ matrix, denoted by L:

$$L = \left[\begin{array}{ccc|cccc} \|p'_1 - p'_1\| & \cdots & \|p'_1 - p'_N\| & 1 & x'_{p_1} & y'_{p_1} & z'_{p_1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \|p'_N - p'_1\| & \cdots & \|p'_N - p'_N\| & 1 & x'_{p_N} & y'_{p_N} & z'_{p_N} \\ \hline 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ x'_{p_1} & \cdots & x'_{p_N} & 0 & 0 & 0 & 0 \\ y'_{p_1} & \cdots & y'_{p_N} & 0 & 0 & 0 & 0 \\ z'_{p_1} & \cdots & z'_{p_N} & 0 & 0 & 0 & 0 \end{array}\right] \tag{4.17}$$

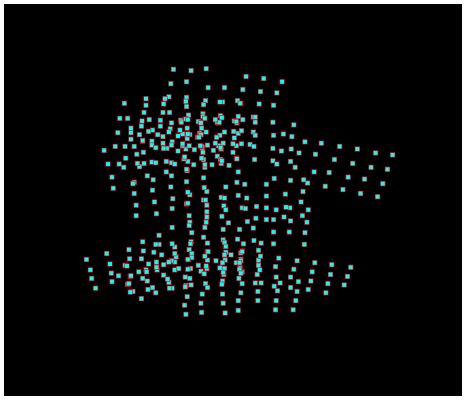
the linear system can be written in the more compact form:

$$L \left[\begin{array}{ccc} w_{1x} & w_{1y} & w_{1z} \\ \vdots & \vdots & \vdots \\ w_{Nx} & w_{Ny} & w_{Nz} \\ \hline a_{1x} & a_{1y} & a_{1z} \\ a_{2x} & a_{2y} & a_{2z} \\ a_{3x} & a_{3y} & a_{3z} \\ a_{4x} & a_{4y} & a_{4z} \end{array}\right] = \left[\begin{array}{ccc} x_{q_1} & y_{q_1} & z_{q_1} \\ \vdots & \vdots & \vdots \\ x_{q_N} & y_{q_N} & z_{q_N} \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \tag{4.18}$$

Each column of coefficients corresponds to the parameters of a TPS, modeling the warping in each of the three spatial dimensions (x, y, z). The lower 3×3 matrix is a transposed version of the 3D affine matrix A of the transformation.



(a) Before calibration (b) After affine calibration



(c) After non-rigid correction

Figure 4.5: Calibration steps.

Mapping the Points

The warping coefficients which map $\{p'\}$ to $\{q\}$ are given by solving Eq. 4.18

$$\begin{bmatrix} \mathbf{w}_x & \mathbf{w}_y & \mathbf{w}_z \\ \mathbf{a}_x & \mathbf{a}_y & \mathbf{a}_z \end{bmatrix} = L^{-1} \begin{bmatrix} \mathbf{x}_q & \mathbf{y}_q & \mathbf{z}_q \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{4.19}$$

where $\mathbf{x}_q = [x_{q_1}, \ldots, x_{q_N}]^T$, $\mathbf{y}_q = [y_{q_1}, \ldots, y_{q_N}]^T$, $\mathbf{z}_q = [z_{q_1}, \ldots, z_{q_N}]^T$ the vectors formed by the coordinates of the points in {q}. In matrix form, the mapping $\{p''\}$

of points $\{p'\}$ is then given by

$$\begin{bmatrix} \mathbf{x''}_p & \mathbf{y''}_p & \mathbf{z''}_p \end{bmatrix} = \begin{bmatrix} C & Q \end{bmatrix} \begin{bmatrix} \mathbf{w}_x & \mathbf{w}_y & \mathbf{w}_z \\ \mathbf{a}_x & \mathbf{a}_y & \mathbf{a}_z \end{bmatrix}, \tag{4.20}$$

where $C_{ji} = \|{p'}_j - {p'}_i\|$ and the j-th row of Q being $[1, {x'}_{p_1}, {y'}_{p_1}, {z'}_{p_1}]$ [48].

Complexity

The system solution requires inversion of the array L. This however has a complexity of $\mathrm{O}(N^3)$ which makes the algorithm very expensive for setups with many calibration points. For such setups Donato et al. [22] provide three ways of dealing with the increased complexity problem; (i) random subsampling of the candidate points, (ii) using only a subset of exact basis functions, (iii) using a full set of approximate basis functions. In our case, N is relatively low, so we were never faced with the need to do any efficiency optimizations. The running time for $N \approx 500$ points was smaller than 10 sec.

Chapter 5

Capturing Calibration Points

In this section we present two methods of obtaining reference calibration points. The first one takes advantage of the direct mapping between the infrared image (IR) and the depth map and employs the checkerboard pattern in order to retrieve corner locations. Concretely, this approach directly depends on having access to the IR image stream. As an alternative to such a prerequisite, we also explore a different method for obtaining points by making use only of the depth map and an easily detectable spherical object.

5.1 Using the Infrared Image

The IR image can be directly mapped to the depth image, except from a narrow band of eight pixels width, that is invalid (zero values) due to the correlation window

used for the triangulation process [60]. We propose to take advantage of this mapping in order to find the 3D positions (in world coordinates) of IR-detected image points, specifically checkerboard corners, and use the derived points as calibration points directly in the depth space. Intuitively, the use of the readily available 3D representation of points is more natural because the errors introduced from the depth mapping are not ignored.

5.1.1 Obtaining 2D Points

The checkerboard pattern is an established method for capturing points during calibration in conventional cameras. By using the same principle for depth cameras, we rely on the same type of points and avoid the need for specialized or custom-designed calibration objects. Furthermore, we can take advantage of well-established algorithms and fairly robust open source implementations for detecting contrast-based corners on the checkerboard pattern. Detection on the monochrome images is done via the Harris corner operator [26], that discovers high gradient changes (edges) in the vertical and horizontal directions.

In Fig. 5.1 we show the interface developed as part of this thesis in order to capture simultaneously IR images (top) and depth maps (bottom) for two Kinects. In order to increase the accuracy of corner points detection in the IR image, the laser emitter is masked during the captures, either by manually covered the Xbox Kinect, or programmatically deactivated for the Windows Kinect device. Figure 5.2 shows sample IR images of the captured checkerboard and their corresponding depth maps.

Figure 5.1: Interface to capture infrared (top) and depth (bottom) images for two Kinects.

5.1.2 Transforming Points to 3D

After localization in an IR image, points are mapped on the corresponding depth image, based on the continuous mapping between them. The depth values on the resulting locations can be made smoother by averaging them over local neighborhoods, i.e. typically on a 9×9 smoothing window. Figure 5.3 shows the detected checkerboard corners (left) mapped to the depth image (center). Each corner p in the 2D depth map is essentially a triplet $p(u, v, z)$. While u and v coordinates are in image space, z is in world space and corresponds to the distance of the 3D point P with

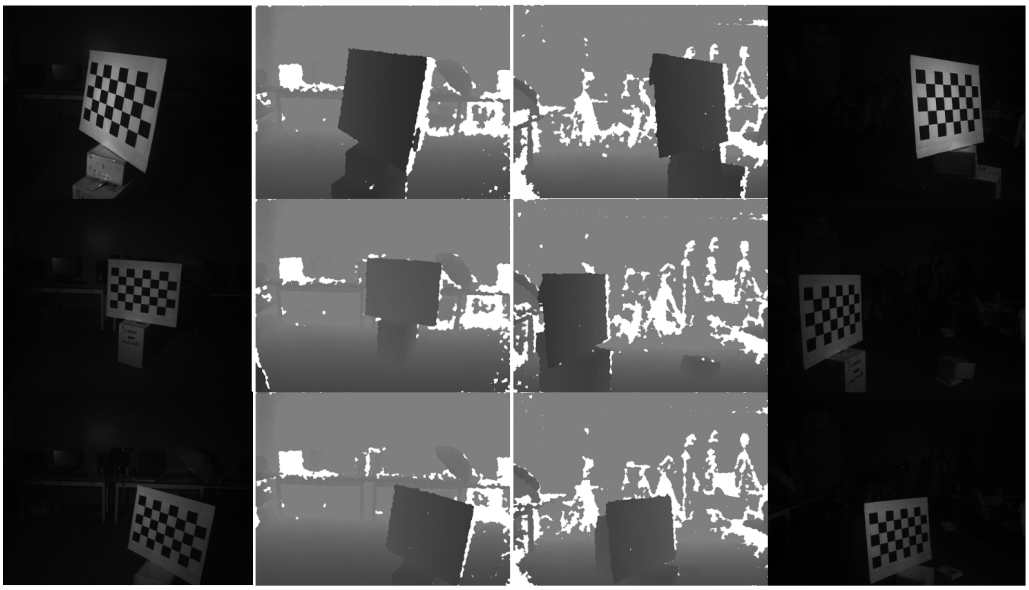


Figure 5.2: Example setup with infrared and depth images captured.

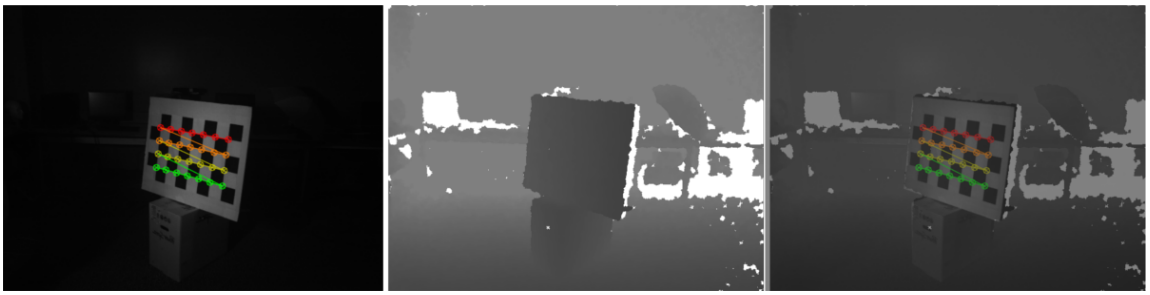


Figure 5.3: Detected 2D points mapped to 3D. Left image shows the mapping (superimposed) and the location of points in both domains.

coordinates (x, y, z) to the camera. As in all cases, we make the assumption that the center of the world coordinate system coincides with the camera origin. To retrieve the actual 3D position of P, we additionally need to map u, v to the corresponding world coordinates x, y.

In order to understand better how this mapping is done analytically, we present

and briefly discuss the pinhole camera model [29]. The model (Fig. 5.4) provides an approximation of the perspective projection of 3D points in space to the 2D image plane. It is based on the assumption of an ideal camera with just a point-sized hole and no lenses so that a single ray can pass through the pinhole. Although such a model is not physically feasible, it nevertheless provides a sufficient approximation for the process and is mathematically convenient [29, 26].

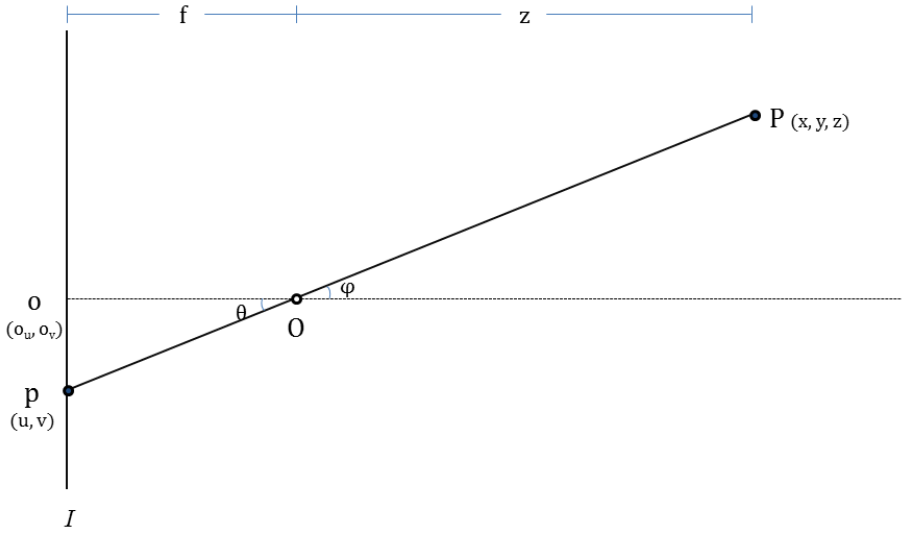


Figure 5.4: Geometry of the pinhole camera model.

From the geometry of Fig. 5.4, since $\hat{\theta} = \hat{\phi}$, we get from similar triangles

$$\frac{u}{x} = \frac{v}{y} = \frac{f}{z}, \tag{5.1}$$

It then follows that

$$u = f_u \frac{x}{z}, \quad v = f_v \frac{y}{z}. \tag{5.2}$$

where f_u and f_v is the focal length in pixels, and are generally different under the general assumption that the pixels are rectangular and thus do not have the same

resolution along u and v. However, the center of the image plane does not necessarily coincide with the point that the Z axis intersects the image plane I [51], and this shift should also be accounted for:

$$\begin{aligned} u &= f_u \frac{x}{z} + o_u \\ v &= f_v \frac{y}{z} + o_v. \end{aligned} \tag{5.3}$$

Equation 5.3 can be rewritten using homogeneous coordinates for p is:

$$p = CP \Rightarrow \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_u & 0 & o_u \\ 0 & f_v & o_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5.4}$$

Matrix C is then precisely the camera or intrinsics matrix which we retrieve by camera calibration. Equations 5.3 and 5.4 are showing how a 3D point projects to the 2D image plane. Recall that our problem is the exact opposite however; from image coordinates (u, v) retrieve real world coordinates (x, y), i.e. from Eq. 5.3:

$$\begin{aligned} x &= (u - o_u) \frac{z}{f_u} \\ y &= (v - o_v) \frac{z}{f_v}. \end{aligned} \tag{5.5}$$

By applying Eq. 5.3, we convert each triplet calibration point (u, v, z) to (x, y, z), where now all coordinates are in reference to the Kinect IR camera center and are expressed in mm units. After projecting all detected points from all views to the 3D space, we create essentially a calibration point cloud for the specific Kinect, that non-uniformly samples points in the camera's 3D space. In Fig 6.7 we present such a point cloud constructed from 17 views with 28 points on the average in each.

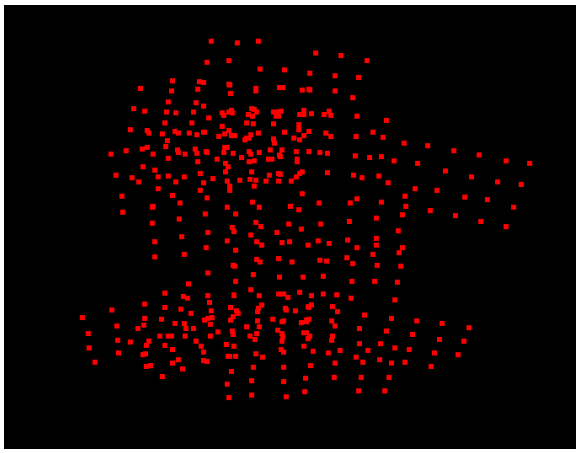


Figure 5.5: Detected checkerboard corners (28) from multiple views (17) converted to 3D point cloud.

5.1.3 Using RGB Instead of IR Camera

Alternatively one might consider using the RGB camera in order to collect points of interest. In that case, the three color component channels can be exploited to distinguish points based on their color features, thus eliminating the need for a calibration board. However, in this case a direct mapping between the color frame and the depth map will not be valid, since the images stem from different sensors. Instead, we must reside to mapping the points to depth values by using the transformation between the IR and RGB, estimated through the Kinect extrinsics calibration process. This transformation is in essence an approximate solution, so a re-projection error will always exist, no matter how small. Another drawback might be the need for a more controlled capturing environment, in order to use highlight specific features or detect certain colors.

5.2 Using the Depth Map Directly

In this section, we explore another promising option for detecting and capturing calibration points by making use only of the depth map. This would be useful in cases where we do not have access to the IR stream or for depth sensors other than Kinect which return only a range map. Detecting points of interest from depth maps is not as straightforward as detecting them in the RGB frames, and features relate primarily to shape than texture. We could use some type of 3D feature extraction (physical or detected such as 3D-Harris or 3D-SIFT) to match with known depth points of interest, though as in the case of the RGB camera, this would require controlled capturing environments. In addition, 3D feature detection matching would be challenging in extreme angles or fail in the case of perpendicular camera positions, since the cameras would face a limited part of the same scene.

To overcome these difficulties, we propose a method by employing a non-specular, spherical object with the intent of being easily detected from all views. By moving a sphere of sufficient size around space, we can produce a fair amount of frames with the object (a ball) in the foreground. Further, detecting the ball in each frame, and using it's known geometry in fitting a simple spherical model, we can select its center as 3D point of interest (5.8). We present a brief outline of the algorithmic steps involved in extracting such points of interest for calibration:

1. Background removal.

2. Detection and parameterization of the sphere using RANSAC based model fitting.

3. If the sphere is not detectable (i.e., low resolution), resampling using Moving-Least-Squares and go back to (2).

4. Return center of sphere.

5.2.1 Background Removal

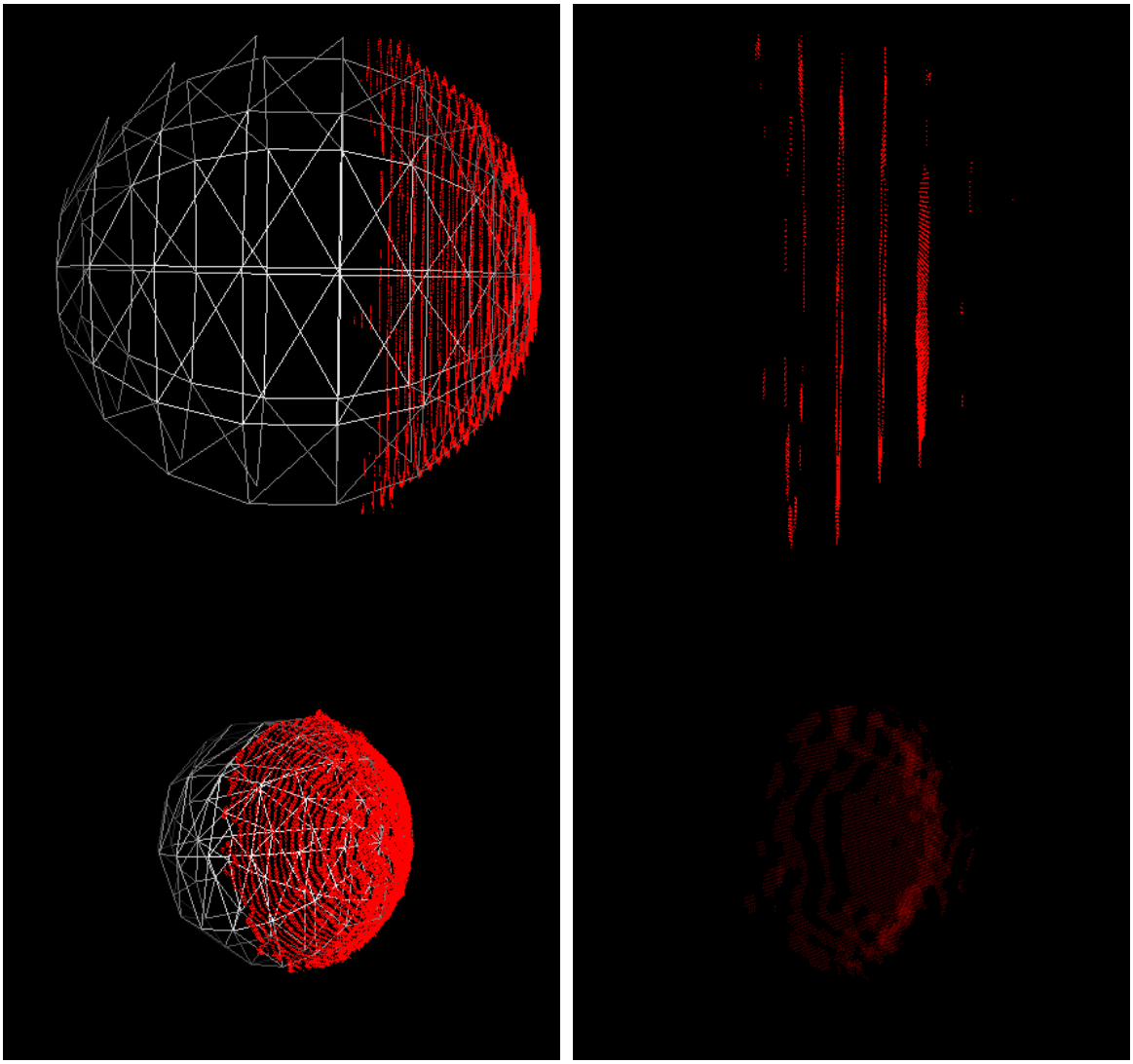
In this step we can distinguish two cases, depending on the way the ball moves around the 3D space:

A person holding the ball In this case manual background removal would be necessary, since apart from the static background, we need to separate the person figure, whose position in each frame varies. For each frame the depth is sampled and manual depth thresholding is performed.

The ball is hanging from a standard point with the use of a cord In order to overcome this cumbersome procedure, we can attach the ball to a cord and let it move freely while we record its positions. In that case, a simple, automatic background subtraction suffices. However, it may be very difficult to control the ball in order to cover the desirable area and the configurations would be very difficult for large setups.

5.2.2 RANSAC-based Model Fitting

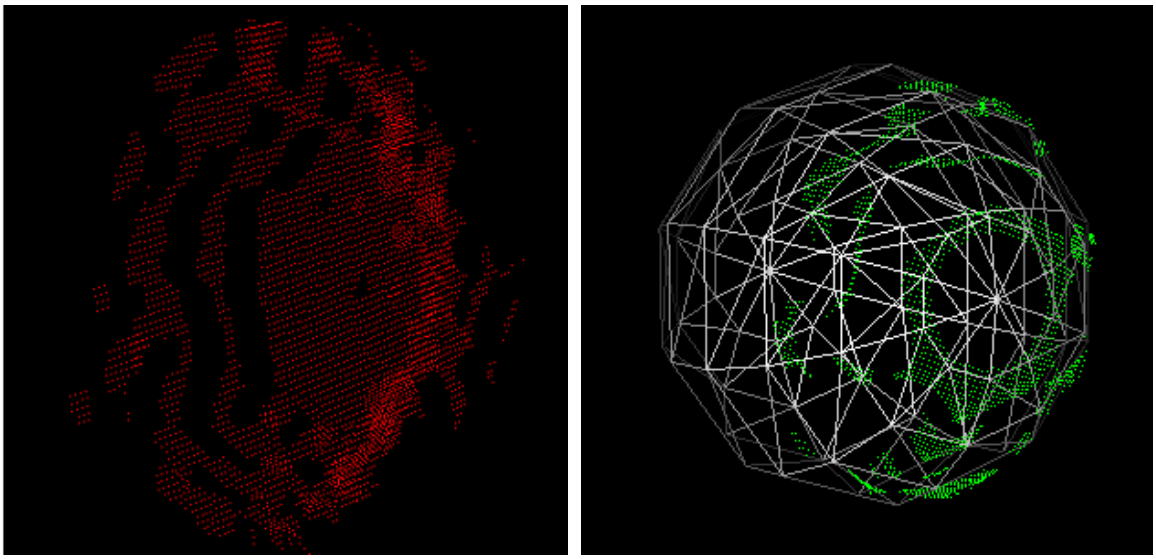
Based on the assumption that the acquired point set will include outliers, we apply a RANSAC ([25], [68], [26])-based fitting of a model for a sphere, in order to



(a) 1 m away from the Kinect.

(b) 3 m away from the Kinect.

Figure 5.6: Depth quantization step: At 3 m the depth levels are not enough for detection.



(a) Not enough points to fit sphere. (b) After MLS resampling.

Figure 5.7: Moving Least Squares to upsample sphere.

retrieve a set of inlier points that can be described by a 3D sphere function. As outliers we consider points that have either remained after background extraction or noisy variations near the surface of the sphere.

5.2.3 MLS Resampling

As we discussed in Chapter 3, the resolution of the sensor decreases dramatically with the distance from the camera. As a result, when the distance of the ball to the sensor is larger than 3 m, there will not be enough surface points for the function to properly fit the depth data (Fig. 5.6). In such cases, we employ the Moving Least Squares algorithm ([47], [43]) to upsample the cloud and retry the fitting (Fig. 5.7).

5.2.4 Center Extraction

Since the sphere implicit function is known, we can easily determine the center of the detected object. In Fig. 5.8 we show the calibration points (sphere centers) for a free moving sphere setup, captured by a Kinect pair.

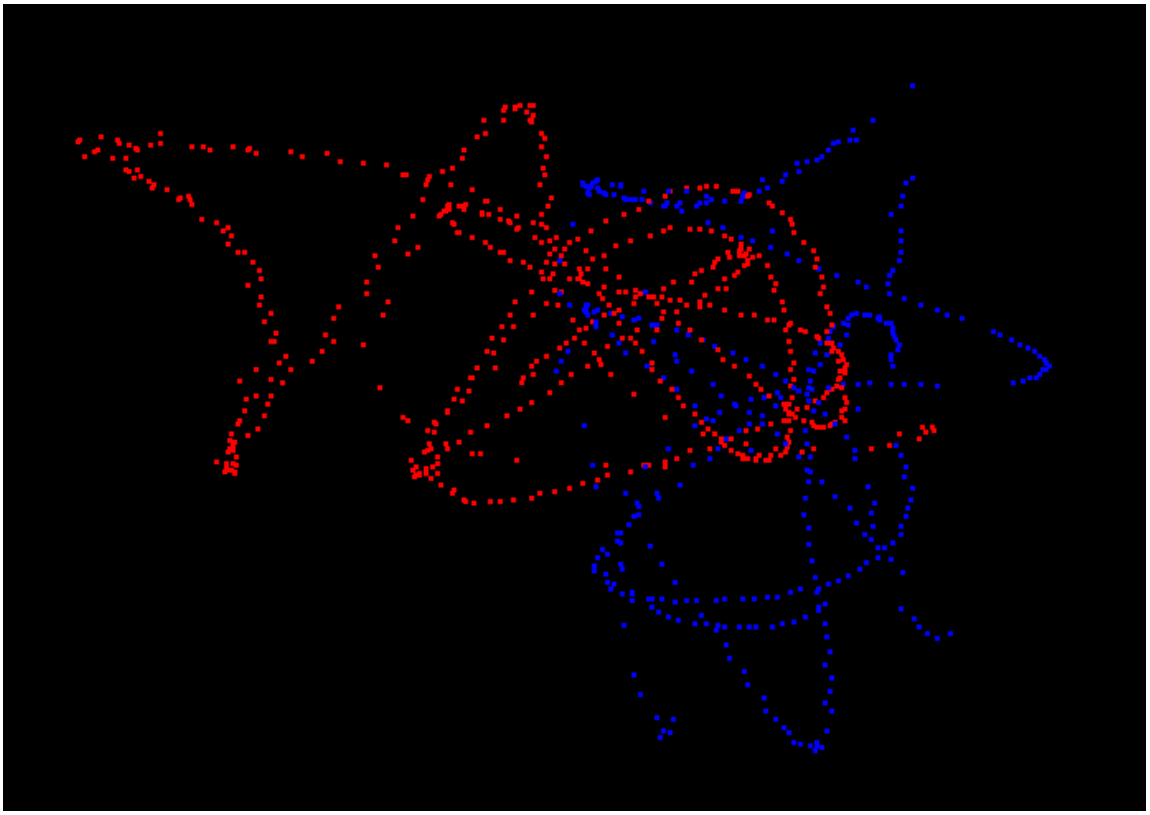


Figure 5.8: Calibration points acquired by using a sphere and the depth map (Red-Right Kinect, Blue-Left Kinect).

Chapter 6

Reconstruction Experiments and Calibration Evaluation

In this Chapter we present experimental results on an application of data registration for object/body scans, using our calibration method and compare it with the most frequently used approach for stereo calibration from two cameras, described in detail in Sec. 3.2.3. Before the qualitative evaluations and analysis of the results, we describe the employed setup for capturing the scenes and any preprocessing performed on the point clouds before registration.

6.1 Experimental Setup

Two important issues that become apparent in setups involving multiple Kinect devices (multi-Kinects) relate to a) connecting the sensors and b) dealing with the

noise introduced by the multiple sensors (interference, invalid values). In this section, we analyze and provide details on how we tackled these connectivity and noise issues.

6.1.1 Kinect Network

The Kinect, as a streaming device needs at least 50% of the USB bandwidth and each additional device must be connected to its own USB hub. Furthermore, if the frames are saved in disk, the multiple streams affect the disk writing speed and may decrease the frame rate. Consequently, connecting two or more sensors in the same system is not a straightforward task. To overcome connectivity issues, we developed a *server-client configuration*, which is adaptable and easily expandable to many sensors. Each Kinect can then be connected to a different computer, with one of the computers, or an additional one, functioning as the server. In Fig. 6.1 we present an instance of the interface during a two-client data acquisition.

The clients can be connected with the server either through a Local Area Network or a Wi-Fi. After the server detects all the connected devices, we can synchronize their initialization and start-time of captures by command-based control through the server. During acquisition, in order to avoid processing overhead and possible frame drop, each Kinect writes a binary file with the raw depth data, a binary file with color data and two text files with frame info. These are local per client and there is no need transmitting them back to the server, especially given their large size.

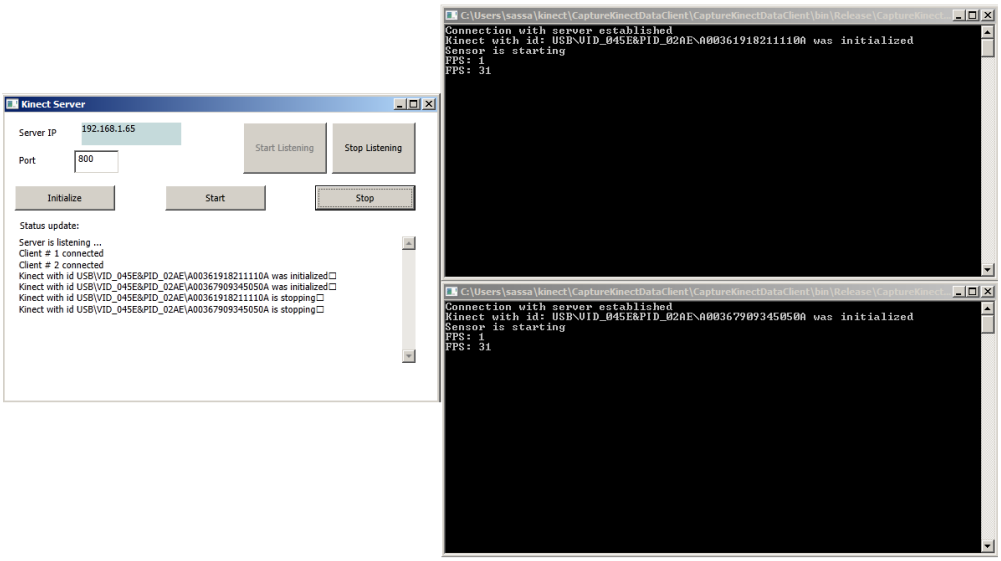


Figure 6.1: Server interface with two connected Kinect clients.

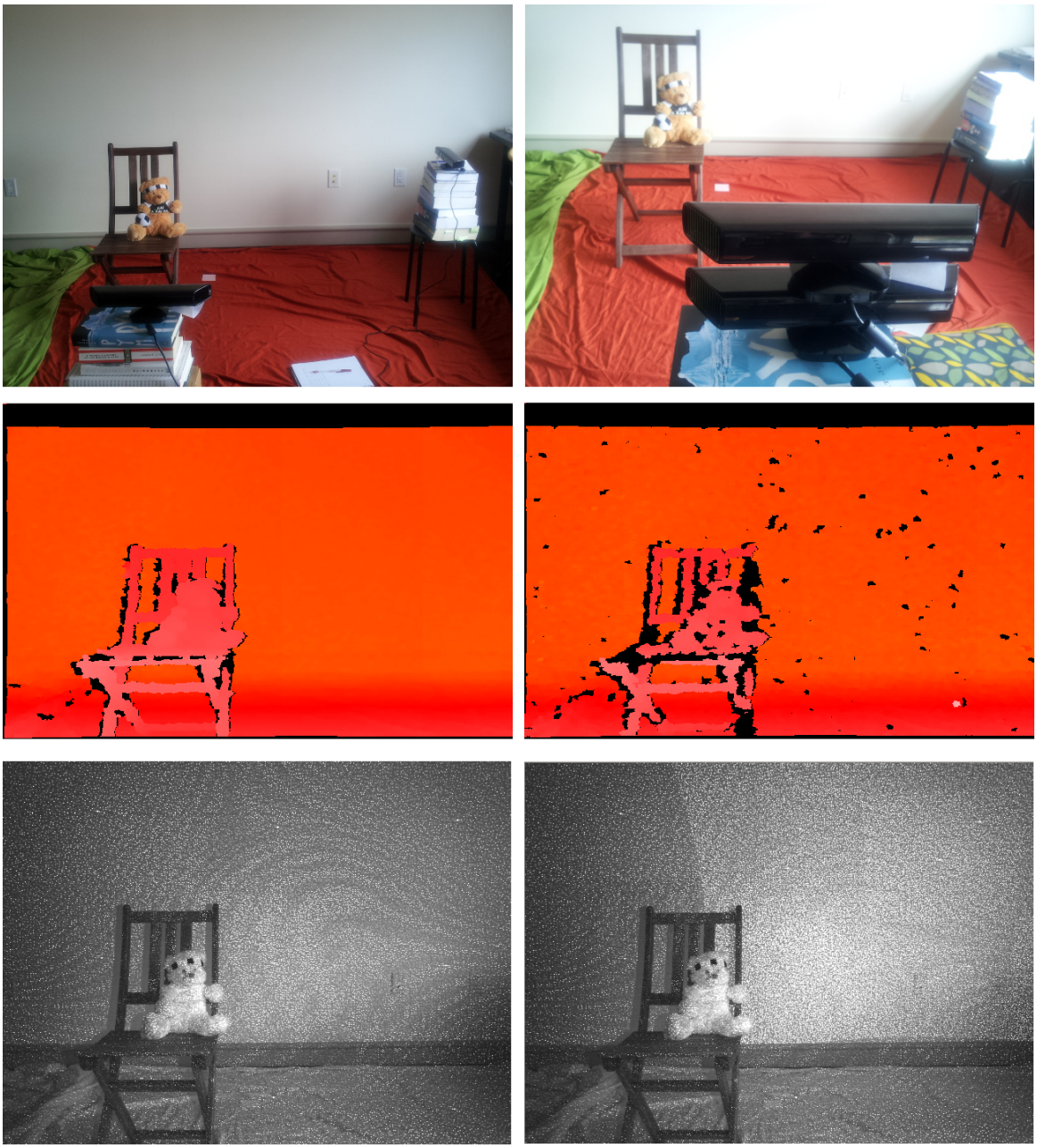
6.1.2 Multiple Kinects Interference

In Chapter 3 we presented in detail the triangulation process through which Kinect infers depth values for a scene. The important point to remember is that the method is based on comparing a stored reference pattern to an infrared-captured image of a scene on which a dot pattern is projected. When more than one sensors project a pattern to the same scene, then each sensor has trouble distinguishing its own and thus noise artifacts and invalid values appear. The amount and effect of noise depends on the positioning of the sensors, relative to the scene and each other. Rafibakhsh et al. [57] did a comparative evaluation of various angles between Kinects in order to minimize the interference and propose a minimum angle of 35 degrees between two sensors for better results. In Fig. 6.2 both the cause and the result of

sensor interference are shown. In the first case (Fig. 6.2a) two sensors are placed orthogonal to each other. Interference in this case is almost non existent since the majority of the projected dots fall on different surfaces. However, in Fig. 6.2b, we present a worst-case scenario interference-wise; the sensors are on top of each other and their projections on the surface are highly overlapping. The IR images on the bottom show the projected pattern as it is perceived by the same sensor. The mixing of the patterns in the second (lower right) case is obvious, and is reflected on the depth map (middle images).

A couple of approaches have been proposed in order to mitigate the interference problem for multi-sensor environments. Kramer et al. [39] use a custom-made hardware shutter used to interrupt the IR emitter and the IR camera alternatively (in turns) in two sensors and keep only the valid frames post-capture. Schröder et al. [62] evaluate the feasibility of options such as directly controlling the laser programmatically and implement a time-multiplexing method with a different kind of shutter (disk).

A different approach is related to adding motion to one or more of the Kinects, which has been shown to improve significantly the performance and reduce noise [14, 50]. By attaching a small motor on the Kinect, a motion blur is introduced which assists in differentiating the patterns. A toy example is shown in Fig. 6.3. The left image simulates dot patterns from two Kinects while the right one is produced by introducing movement on one of the two devices. With the added motion blur, the static Kinect can distinguish its own dots, while the moving one is also discriminative to its own pattern, as the dots move simultaneously.



(a) Kinects are placed orthogonally (b) Kinects on top of each other

Figure 6.2: Amount of interference based on two Kinect sensors' configuration: a) placed orthogonally; the amount of noise from interference is miniscule b) placed on top of each other and project dotted pattern in the same scene; interference noise is non negligible.

Initially, while exploring possible setups, we also experimented with the motion as compensation for interference, based on a specific configuration with multiple (up to four) devices, where every other Kinect would change its pitch angle (up and down). Angles were changed every 20 seconds, although Kinect specs dictate that it should remain still for 20 seconds after 15 tilts. However, this introduced a major complexity in the system, since the devices needed frame-wise re-calibration, at the expense of virtually no improvement, since the effect of degradation on the final reconstructions was not serious. This setup was eventually not incorporated in out framework, due to the heavy trade-off between complexity and data improvement.

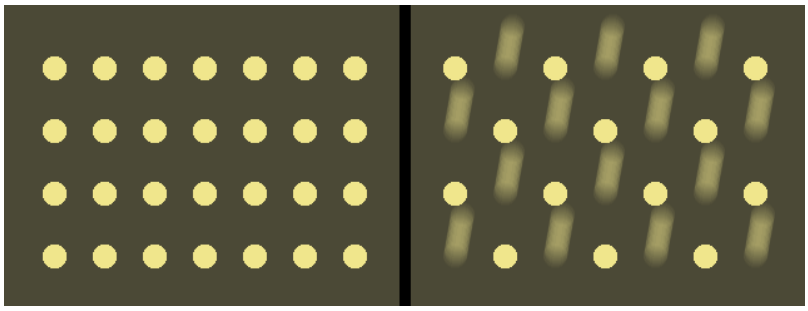


Figure 6.3: Dot pattern interference with and without enforced motion blur.

6.2 Preprocessing for Data Registration

To get from two Kinect captures to a single, registered scene, certain preprocessing steps are required, related to separating the foreground data, producing a point-cloud representation and registering the color data for a 3D/texture final representation. Next we separately discuss these processes, under the framework of obtaining point clouds for registration.

6.2.1 Background Removal

In applications where we focus on an object of interest, automatic background removal is necessary to isolate that object. Manual background subtraction is possible, but it becomes laborious and cumbersome for a large amount of data. In the case of depth frames, we are not only restricted to color for object indication, and the additional stream of depth values can be used to segment the object of interest. In [41], where an RGB and depth object database is presented, a bounding box initially removes most of the background and a RANSAC plane fitting determines a third bounding dimension for depth selection. In [10], part of a method for gesture recognition, background subtraction is done by automatic thresholding in the depth image and removing values based on a valley-peak rule. A learning approach with linear classifiers and HOG descriptors was used for extracting humans from background [28], while in Stone et al. [64], a background model is created from a number of frames.

In our processing, we want to isolate a distinct object or group of objects in order to perform calibration-based data registration from different sensors, and validate the proposed calibration approach. Our background subtraction for separating the object of interest resembles closely the one in Stone et al. [64], by making use of a background model (Fig 6.4). Subtraction is applied in the 2D image, since this way we can take advantage of the depth info as well as some 2D image processing techniques. In detail, the algorithm steps, which are visualized in Fig. 6.5 are the following:

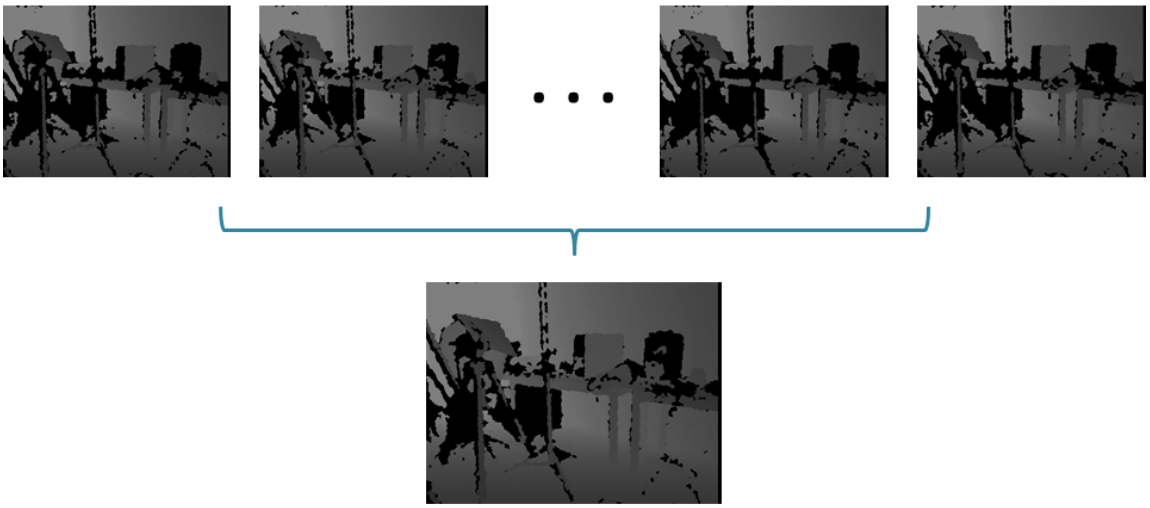


Figure 6.4: Frames for building a background depth model.

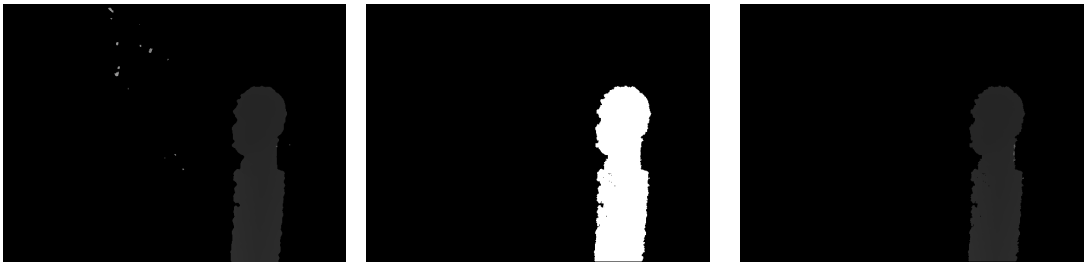
Background selection

We use M frames (usually $M \approx 50$) to construct a model for the background (Fig 6.4). Multiple frames are needed in order to alleviate the frame noise variation. The model is compared with each target frame; if model$[i, j] \leq$ frame$[i, j]$, then frame$[i, j] = 0$ and pixel$[i, j] \in B$ is considered background (Fig 6.5c).

Filtering

The initial subtraction is usually not enough, e.g., see in Fig 6.5c, since it may keep small-scale, isolated artifacts and depth noise, due to the point-wise comparisons. A median filter with a 5×5 support window is further applied to retain larger scale objects (Fig 6.5d).

(a) Original RGB image (b) Original depth image (c) Background removal



(d) Median filtering (e) Largest contour mask (f) Final image

Figure 6.5: Background subtraction steps.

Contouring

The object of interest is found through contouring; after detecting all image contours, we select the largest one (Fig 6.5e) assuming it corresponds to the main scene object. The contour mask is applied on the original image to return the depth values (Fig 6.5f).

6.2.2 Converting Depth to Point Clouds

In Section 5.1.2 we have detailed how to transform a depth frame to its corresponding 3D point cloud. We briefly outline the basic formulation here. We assume that we have a 2D point in the depth frame, with image coordinates u, v and depth

value z, in world (camera) coordinates (mm units), and use it to retrieve (x, y) coordinates:

$$(x, y) = \left(\frac{u - o_u}{f_u}, \frac{v - o_v}{f_v} \right) * z,$$

where f_u and f_v is the focal length in each direction and (o_u, o_v) the IR camera's center of projection. These four parameters are obtained through intrinsic calibration of the IR camera, described in Chapter 3. In Fig. 6.6 we show the point cloud that was produced from the corresponding depth map, for a scene with different objects.

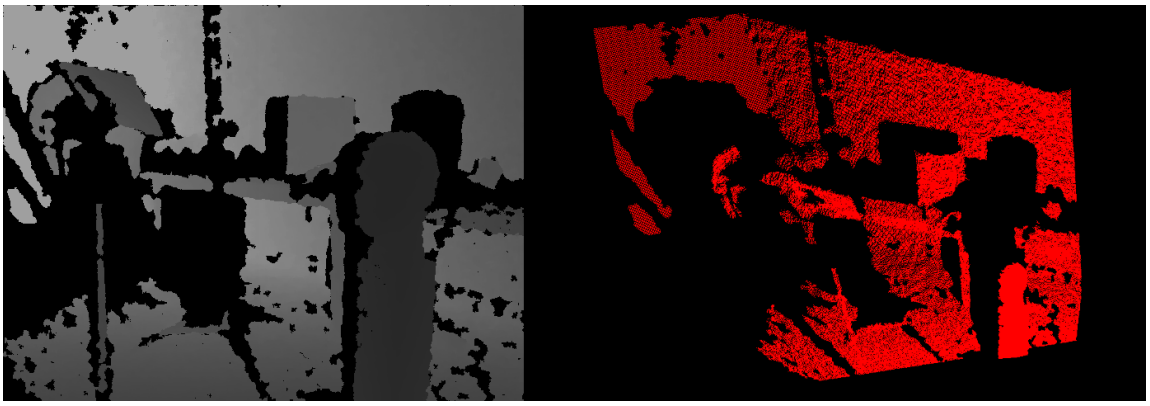


Figure 6.6: Depth map converted to point cloud.

6.2.3 Coloring the Point Clouds

The registration of color values with point clouds, amounts to assigning an rgb value to the each 3D point (x, y, z), based on the underlying image values of (u, v) and the known IR-color mapping. This is useful for visualizing 3D point clouds, and reconstructed meshes, associated with a texture value that captures the color of each point in 3D. We assume that we have stereo-calibrated the IR and RGB camera and

know the transformation between the two coordinates systems. In order to determine an RGB value for each point, we use the following steps:

1. **Transform point cloud** from IR space to RGB space using the affine matrix acquired during RGB-IR stereo calibration.

2. Project the transformed **point cloud to the 2D color image space**, using the RGB camera intrinsic parameters. Assuming (x, y, z) is a 3D point in color camera space, then its 2D projection (u, v) is given by:

$$u = (x/z) * f_u + o_u, \quad v = (y/z) * f_v + o_v,$$

   where $\{f_u, f_v, o_u, o_v\}$ are the intrinsics of the RGB camera.

3. Color the (x,y,z) point with the RGB value at (u,v).

In Fig. 6.7 we visualize an example of a 3D scene with the color values of the points superimposed, i.e. a colored point cloud. This composite representation captures the relative positions (scene structure) and the visible-light variations (scene texture) that identify objects by color or texture values.

6.3 Registration Results and Comparison

In this Section we present our main validation results for the effectiveness of the proposed calibration method and framework. In effect we aim to support our claims that a) calibration based on 3D points is more accurate and efficient than conventional stereo calibration methods, b) choice of a small number of points directly in

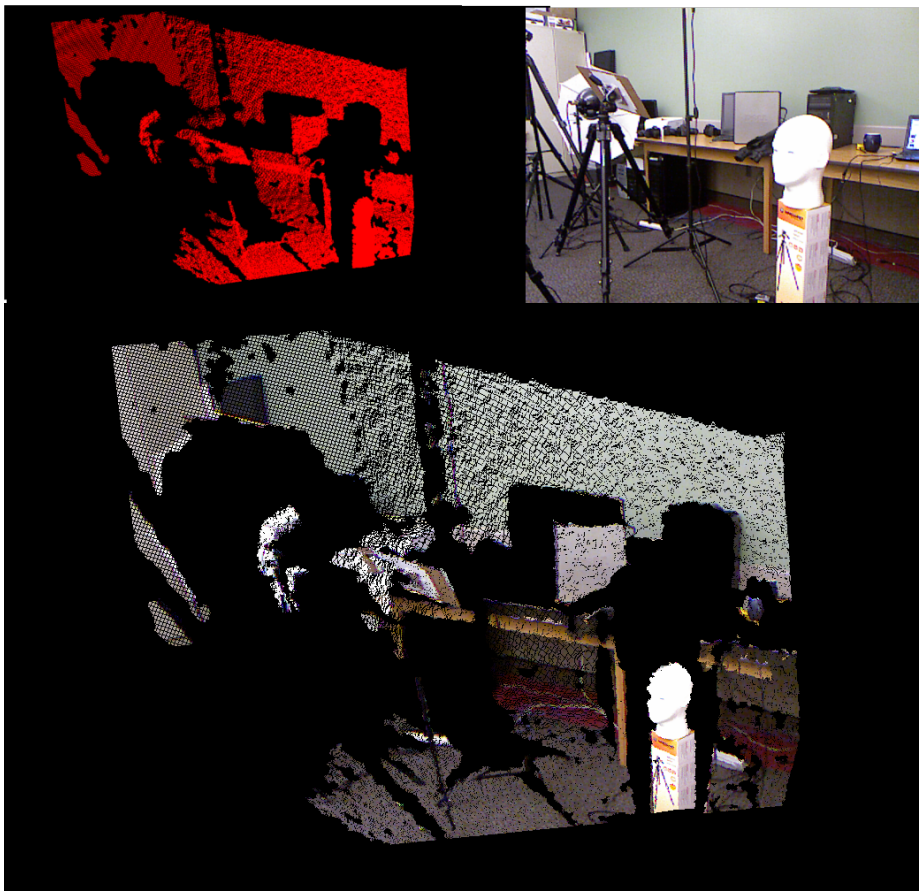


Figure 6.7: Coloring of point cloud through mapping of the RGB frame.

the 3D space is practical and more accurate with respect to sensor errors, c) parameters and camera positions should always be calibrated and accounted based on the specific devices and acquisition set-ups.

The target goal is to achieve two-scan body and object registration with data from two Kinect sensors. Unfortunately, due to the lack of ground truth data or standard applications/databases for quantitative evaluations, established evaluation protocols and metrics, it is hard to establish a statistically meaningful quantitative

comparison. We reside on our own captures, set-ups and data, for which a ground truth point cloud is not feasible, and evaluate registration results qualitatively using various visual assessment metrics, i.e., perceptual quality, naturalness in appearance, preservation of original object features, introduction of non-existing features and artifacts (i.e. edges, contours, asymmetry, surface smoothness).

6.3.1 Data, Methods, Setups, Visuals

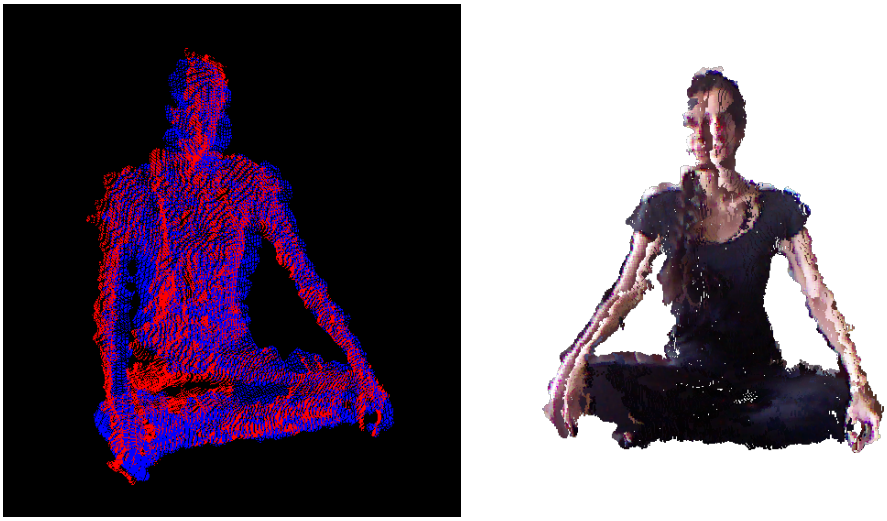
Our main axis of comparison is improvements against the conventional stereo camera calibration method (SC) that most of the current multi-Kinect approaches use (Chapter 2, Section 3.2.3). We use a variety of objects and two human subjects in different body poses and demonstrate the merit of the different components of our calibration method, based on 3D depth correspondences: a) a global affine transform on the data (GA), b) the non-rigid Thin Plate Splines (TPS) which includes an affine component, and c) an initial global affine calibration with non-rigid correction (GA+TPS). For validation we used two different, Kinect pair set-ups: a) positioned at 120° so a partial scene overlap is achieved and b) positioned orthogonal at 90° in which case the sensors view disjoint surfaces. In most comparisons we present reconstruction results as a composite point cloud, with the different methods color-coded (see Table 6.3.2) and a reconstructed scene or object of interest with color values superimposed.

Red	Cloud captured from reference camera A (cloud A)
White	Cloud captured from camera B (cloud B) with no transformation
Blue	Cloud B transformed based on conventional camera calibration
Green	Cloud B transformed based based on global affine transformation and non-rigid correction

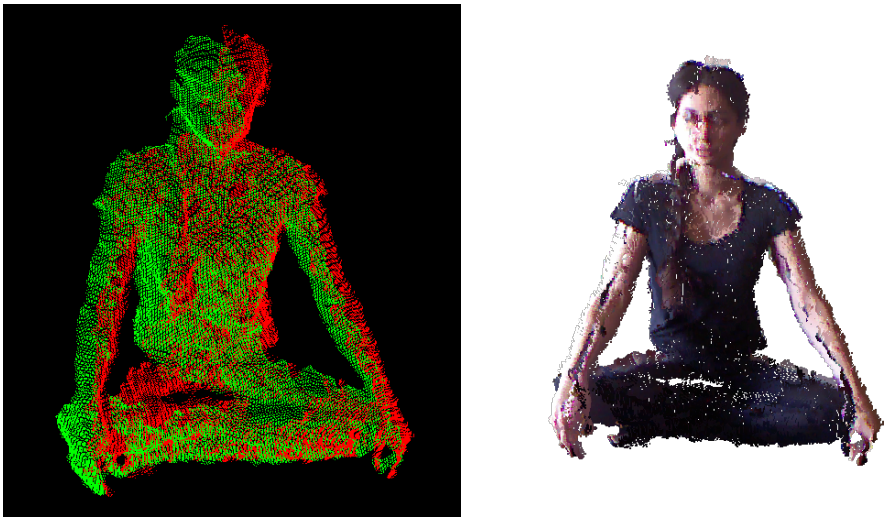
Table 6.1: Color-coded point clouds

6.3.2 Visual Registration Comparisons

Figure 6.8: Uncalibrated point clouds.



(a)



(b)

Figure 6.9: (a) Conventional and (b) our registration results.

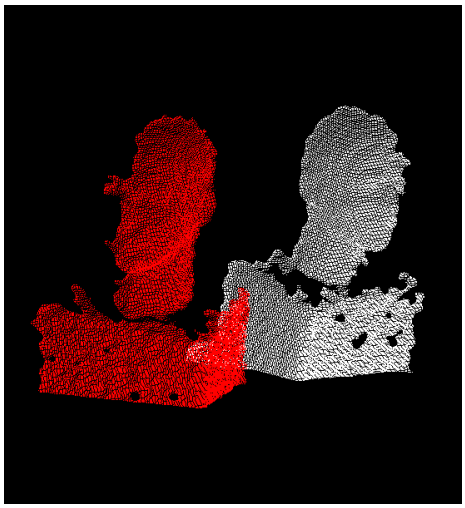
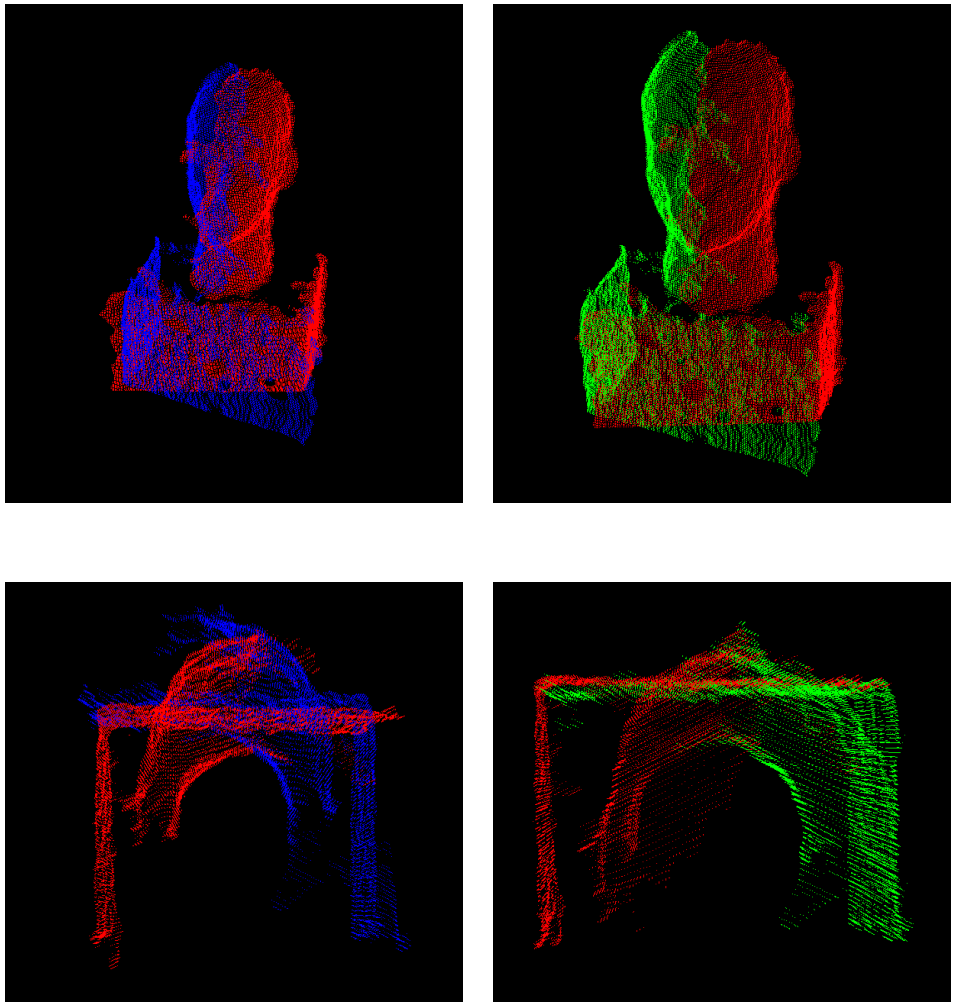


Figure 6.10: Uncalibrated point clouds.

6.3.3 Qualitative and Comparative Analysis

Below we provide a set of qualitative observations that point to specific figure examples.

Proposed method and stereo calibration The proposed method provides improved point-cloud and color registration scans in all cases with better alignment and smoother surfaces and boundaries (see Figures 6.11, 6.16). This observation is consistent across many trials and captures in the sense that by using a default number of parameters and the described intrinsic and extrinsic calibration procedures, the superiority of the depth based calibration method was not undermined. This is also evident by the color reconstructions, for which the conventional method results in more surface artifacts and un-natural looking texture merges (Fig. 6.9).

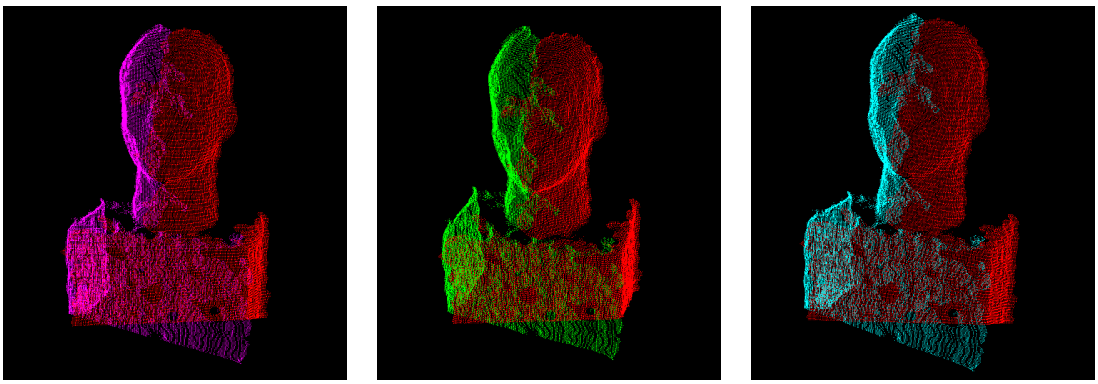


(a) (b)

Figure 6.11: (a) Conventional and (b) our registration results.

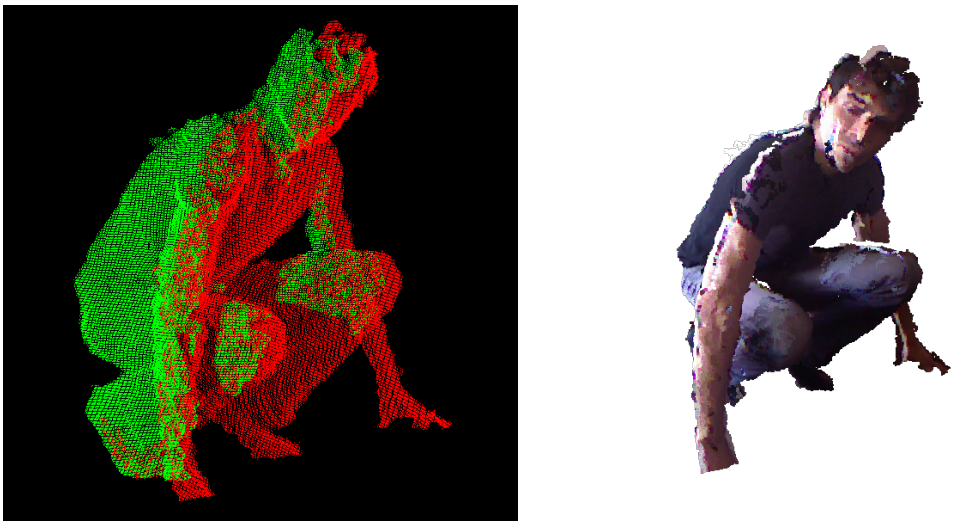
(a) (b)

Figure 6.12: Colored registered point clouds using (a) conventional calibration and (b) our method.

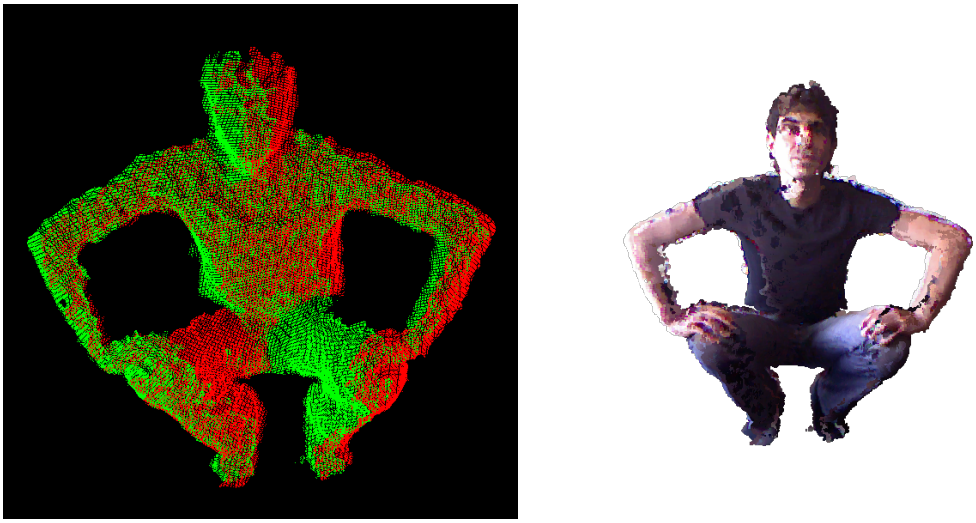


(a) GA (b) GA+TPS (c) TPS

Figure 6.13: Variance of the proposed method

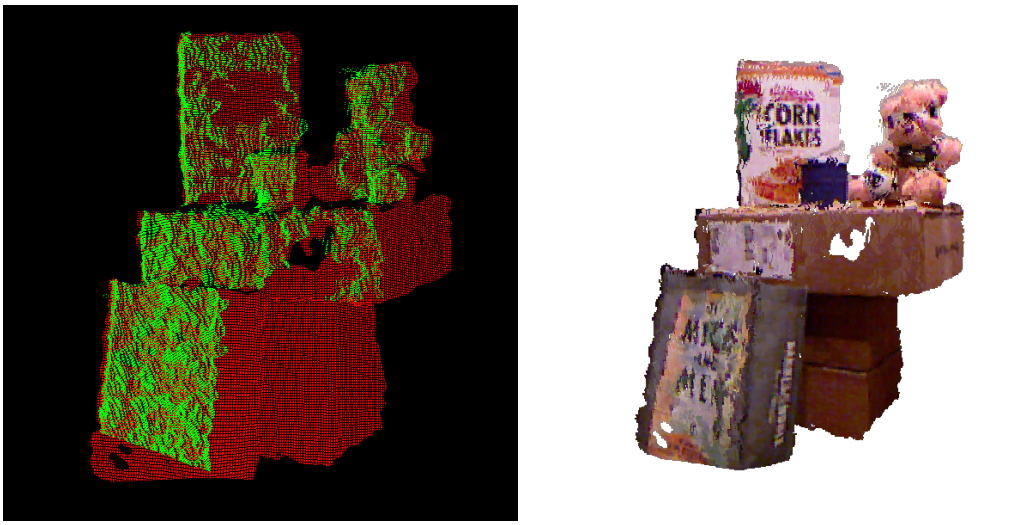


(a)

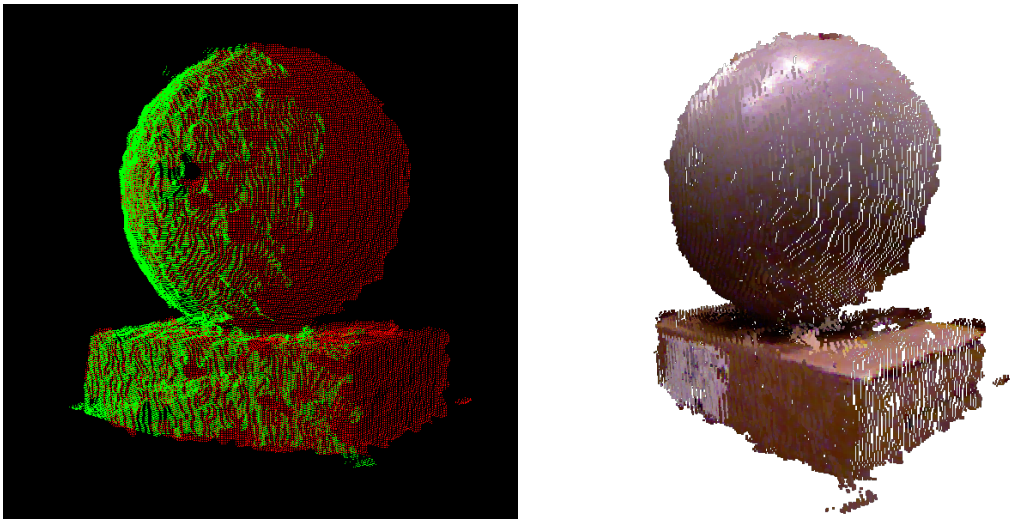


(b)

Figure 6.14: Registration using our method for different poses (a) and (b).



(a)



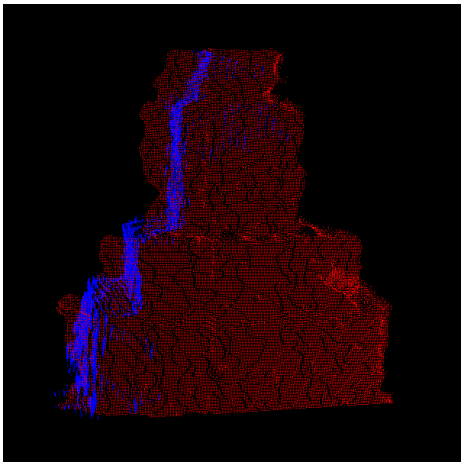
(b)

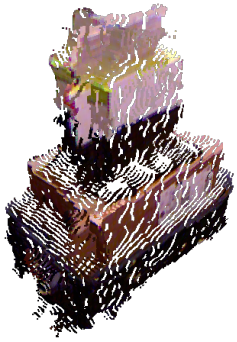
Figure 6.15: Registration using our method for different scenes (a) and (b).

(a)

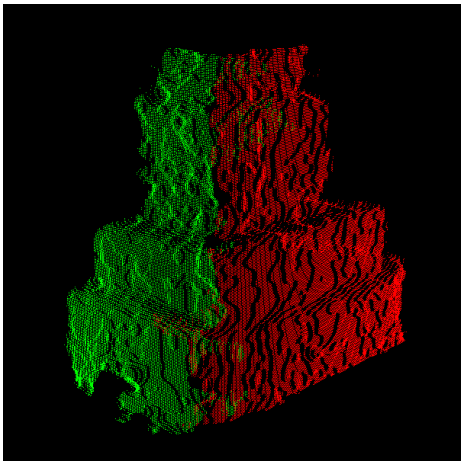
(b)

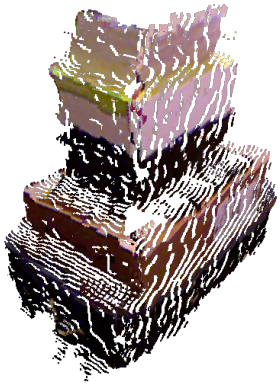
Figure 6.16: Registration results of (a) conventional and (b) our method in cloud with very little overlap.





(a)





(b)

Figure 6.17: Registration results of (a) conventional and (b) our method in clouds with almost no overlap.

Registration under orthogonal views In the orthogonal view scenario, the performance of the proposed calibration method is impressive (Figs. 6.16, 6.17). On the contrary, the traditional stereo calibration, operating on the non-overlapping views, aligns the scans with limited or no overlap. As a result, the registration output is in essence non-usable from methods based on point-to-point correspondence (e.g. correction through ICP). *The conventional method, as opposed to the proposed one, cannot be used for calibration of a registration process from orthogonal, or extreme views!*

Color and point cloud reconstructions In cases where the point cloud reconstruction result looks very accurate, balanced and the scans aligned, the artifacts that appear on the reconstruction with color are ghost-effects, i.e. the color values create an illusion of inconsistency due to subtle surface point cloud variations (Fig.6.14). Such effects may be fixable during an actual mesh reconstruction from the composite point cloud.

Variants of the proposed method In many cases the results from the variants of the method (GA),(TPS),(GA+TPS) look very similar (Fig. 6.13). This can have many probable causes and explanations. In theory the TPS model will always result in optimum results, since it already accounts for an affine component. Any non-rigid refinement will only decrease the distance and error of individual control points. Moreover, since no quantitative evaluations or ground truth exist we cannot be positive on the error-wise performance of each variant.

Chapter 7

Conclusion

In this thesis we outlined a novel method for non-rigid registration of depth measurement acquisition from multiple Kinect sensors. We motivated this work by describing the new framework and requirements brought by the specifications of this new sensor along with the unique aspects of a relevant sensor-dependent, depth acquisition error model.

We showed that the affine model used for calibration of conventional cameras does not suffice and a new model is required to account for the distance-dependent, non-uniformly distributed sensor difference error across pairs of simultaneously captured points. We proposed modeling these variations and depth error inconsistencies using non-rigid deformations, and an associated calibration framework based on affine and non-affine transformations. Object and scene registration results from Kinect pairs were shown to be superior, especially under extreme acquisition angles, in terms of consistency, continuity and alignment of the resulting composite point clouds. The

method depends on establishing intra-sensor point correspondences, for which, as opposed to conventional methods, we proposed two different methods for locating points directly on the 3D space: a) based on reconstructing 3D coordinates from 2D infrared points and b) using a rigid, known object and its estimated location in 3D space. Empirically, the first method is more suitable, due to not suffering from depth measurement inconsistencies in large distances from the sensor. This, however, depends on the specific setup and configuration.

7.1 Future Work

This preliminary study highlighted the potential of using multiple Kinect sensors for capturing full-angle, 3D scans of objects of interest such as human bodies. This can be further generalized to more than two sensors as well as dynamic scanning and modeling across time. At the same time, the results of this work identified a number of short-comings of the current methods in the literature, many of which depend on empirical observations and practical considerations. From the point of view of potential applications, the results can be integrated to methods for shape modeling, motion analysis and recognition, dynamic 3D reconstruction etc.

The potential of the proposed methods, the remaining shortcomings and the targeted applications dictate a number of useful extensions for this work. These relate to refining the selection and identification of landmark calibration points and the generalization to multiple sensors (spatial integration) and dynamic acquisition (temporal integration). In the following we provide brief outlines of examples for

possible extensions and future work.

Additional point retrieval methods

We have presented and compared two methods for selecting calibration points. Since both the initial transformation estimation, as well as the proposed non-rigid extension depend significantly on the distribution and correspondence accuracy of those points, we will look into novel and more robust methods for identifying and establishing depth correspondences. This can involve both sophisticated sampling schemes based on the geometry of the calibration objects and calibration designs adapted to the proposed method. For example, the RGB camera could be used in conjunction with a colored stick in order to detect continuous points in the color frames and map them to get their corresponding depth. However, one should always keep in mind the requirements for a practical system with minimum complexity and user interaction.

Point selection refinement

As was described in Chapter 4, finding the coefficients of the Thin Plate Splines transform can be a very costly procedure when the number of points is large. An approach towards reducing that cost would be to experiment with techniques for reducing the dimensionality of the point sample space in order to get an optimal cost-performance trade-off. A straightforward step would be to apply clustering techniques, e.g., K-means with variations for K, and pick the resulting cluster centers as our chosen points. A more formal approach though would be applying dimensionality reduction techniques like Principal Component Analysis (PCA) to reduce the final number of matching points.

An additional direction would be to explore outlier detection techniques (e.g. like RANSAC) to isolate a set of consistent matches across frames.

Global refinement for more than two sensors

The proposed methods and presented case-study results apply to a dual-camera system. If more cameras are to be included in the configuration, calibration should then be done on a pair-wise manner, i.e., register each camera to a reference one. This however, may be susceptible to cumulative errors, due to the fact that each camera pair is treated independently. This can be compensated for by introducing some form of global refinement of the calibration parameters.

A logical next step would be to expand our method for multi-camera setups (in this context by multi- we mean more than two). There are two ways towards generalizing to multi-camera setups: a) apply a Bundle Adjustment algorithm, modified for depth cameras, as a post-processing step to the initial pair-wise calibration step, b) apply a global calibration method by optimizing with respect to all configuration parameters, in order to simultaneously find the (relative) position of all cameras on the scene. The latter would probably prove more difficult, since this would require an optimization function with multiple unknowns, and thus possibly local minima, and simultaneously retrieve multiple point correspondences from all the available views.

From the point of view of the actual sensor, we anticipate that with the widespread use of the Kinect as a research data acquisition device, future releases and designs will have to acknowledge and cope with issues that are up to now overlooked, such

as data depth resolution and error models, resilience to noise and interferences, data fusion and sensor synchronization. By adapting the ideas of this work to any advancements towards those directions we will be moving towards more efficient (computationally), more robust (to scene or sensor settings) and more accurate (in terms of registration/reconstruction error) methods for multi-sensor, multi-angle, multi-capture dynamic object scans.

Bibliography

[1] Microsoft Kinect for Windows SDK. `www.microsoft.com/en-us/kinectforwindows/`. Accessed: 04/05/2013.

[2] OpenCV. `http://opencv.org/`. Accessed: 03/13/2013.

[3] Openkinect. `openkinect.org/wiki/Main_Page`. Accessed: 04/05/2013.

[4] OpenNI: the standard framework for 3D sensing. `www.openni.org`. Accessed: 03/27/2013.

[5] D. Alexiadis, D. Zarpalas, and P. Daras. Real-time, full 3D reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15(2):339–358, 2013.

[6] M. Andersen, T. Jensen, P. Lisouski, A. Mortensen, M. Hansen, T. Gregersen, and P. Ahrendt. Kinect depth sensor evaluation for Computer Vision applications. Technical Report ECETR-6, Department of Engineering, Aarhus University (Denmark), 2012.

[7] K. Berger, K. Ruhl, M. Albers, Y. Schroder, A. Scholz, J. Kokemuller, S. Guthe, and M. Magnor. The capturing of turbulent gas flows using multiple Kinects. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1108–1113. IEEE, 2011.

[8] K. Berger, K. Ruhl, C. Brümmer, Y. Schröder, A. Scholz, and M. Magnor. Markerless motion capture using multiple color-depth sensors. In *Proc. Vision, Modeling and Visualization (VMV)*, pages 317–324, 2011.

[9] J. Biswas and M. Veloso. Depth camera based indoor mobile robot localization and navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1697–1702. IEEE, 2012.

[10] K. Biswas and S. K. Basu. Gesture recognition using microsoft kinect®. In *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pages 100–103. IEEE, 2011.

[11] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

[12] J.-Y. Bouguet. Camera calibration toolbox for MATLAB. 2004.

[13] N. Burrus. Kinect RGB demo. *Manctl Labs*.

[14] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proc. ACM Annual Conference on Human Factors in Computing Systems*, pages 1933–1936. ACM, 2012.

[15] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 67–76. ACM, 2001.

[16] Y.-J. Chang, S.-F. Chen, and J.-D. Huang. A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities*, 32(6):2566–2570, 2011.

[17] J. Chow, K. Ang, D. Lichti, and W. Teskey. Performance analysis of a low-cost triangulation-based 3D camera: Microsoft Kinect system. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX-B5:175–180, 2012.

[18] Y. Cui, W. Chang, T. Nöll, and D. Stricker. KinectAvatar: Fully automatic body capture using a single Kinect. In *Asian Conference in Computer Vision-ACCV 2012 Workshops*, pages 133–147. Springer, 2013.

[19] Y. Cui, S. Schuon, S. Thrun, D. Stricker, and C. Theobalt. Algorithms for 3D shape scanning with a depth camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1039–1050, 2012.

[20] Y. Cui and D. Stricker. 3D body scanning with one Kinect. In *2nd International Conference on 3D Body Scanning Technologies*, pages 121–129, Lugano, Switzerland, 2011.

[21] Y. Cui and D. Stricker. 3D shape scanning with a Kinect. In *ACM SIGGRAPH 2011 Posters*, page 57. ACM, 2011.

[22] G. Donato and S. Belongie. Approximate thin plate spline mappings. *Proc. European Conference on Computer Vision (ECCV)*, pages 13–31, 2002.

[23] M. Draelos, N. Deshpande, and E. Grant. The Kinect up close: Modifications for short-range depth imaging. In *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 251–256, 2012.

[24] J. Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive Theory of Functions of Several Variables*, pages 85–100. Springer, 1977.

[25] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[26] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall, 2002.

[27] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[28] V. Gulshan, V. Lempitsky, and A. Zisserman. Humanising GrabCut: Learning to segment humans using the Kinect. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1127–1133. IEEE, 2011.

[29] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, volume 2. Cambridge Univ. Press, 2000.

[30] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1106–1112. IEEE, 1997.

[31] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25, 2010.

[32] M. Hernandez, J. Choi, and G. Medioni. Laser scan quality 3D face modeling using a low-cost depth camera. In *Proc. 20th European Signal Processing Conference (EUSIPCO)*, pages 1995–1999. IEEE, 2012.

[33] D. Herrera C, J. Kannala, and J. Heikkilä. Accurate and practical calibration of a depth and color camera pair. In *Computer Analysis of Images and Patterns*, pages 437–445. Springer, 2011.

[34] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[35] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. 24th Annual ACM Symposium on User Interface Software and Technology*, pages 559–568. ACM, 2011.

[36] J.Y.Bouguet. Matlab calibration tool. `www.vision.caltech.edu/bouguetj/calib_doc/`. Accessed: 03/13/2013.

[37] A. Kar. Skeletal tracking using Microsoft Kinect. *Methodology*, 1:1–11, 2010.

[38] K. Khoshelham. Accuracy analysis of Kinect depth data. In *ISPRS Workshop Laser Scanning*, volume 38, page 1, 2011.

[39] J. Kramer, M. Parker, D. Herrera, N. Burrus, and F. Echtler. *Hacking the Kinect*. Apress, 2012.

[40] G. Kurillo, Z. Li, and R. Bajcsy. Wide-area external multi-camera calibration using vision graphs and virtual calibration object. In *Proc. 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–9, 2008.

[41] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[42] K. Lai, J. Konrad, and P. Ishwar. A gesture-driven computer interface using kinect. In *Proc. IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 185–188, 2012.

[43] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981.

[44] B. Lange, C.-Y. Chang, E. Suma, B. Newman, A. S. Rizzo, and M. Bolas. Development and evaluation of low cost game-based balance rehabilitation tool using the microsoft kinect sensor. In *Proc. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1831–1834, 2011.

[45] B. Lange, S. Rizzo, C.-Y. Chang, E. A. Suma, and M. Bolas. Markerless full body tracking: Depth-sensing technology within virtual environments. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, number 1. NTSA, 2011.

[46] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*, volume 161. SIAM, 1974.

[47] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation of the American Mathematical Society*, 67(224):1517–1531, 1998.

[48] J. Lim and M.-H. Yang. A direct method for modeling non-rigid motion with thin plate spline. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1196–1202, 2005.

[49] M. Lourakis and A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm. Technical Report 340, Institute of Computer Science-FORTH, Heraklion, Crete, Greece, 2004.

[50] A. Maimone and H. Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *Virtual Reality Workshops (VR), 2012 IEEE*, pages 51–54. IEEE, 2012.

[51] A. Majumder. Camera calibration. `www.ics.uci.edu/~majumder/vispercep/cameracalib.pdf`. Accessed: 03/24/2013.

[52] R. Mojtahedzadeh. Robot obstacle avoidance using the Kinect. Master's thesis, KTH Royal Institute of Technology, School of Computer Science and Communication, 2011.

[53] MSDN. Color stream. `msdn.microsoft.com/en-us/library/jj131027.aspx`. Accessed: 03/13/2013.

[54] MSDN. Kinect for Windows sensor components and specifications. `msdn.microsoft.com/en-us/library/jj131033.aspx`. Accessed: 03/13/2013.

[55] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proc. British Machine Vision Conference (BMVC)*, pages 101.1–101.11, 2011.

[56] W. Qin, Y. Hu, Y. Sun, and B. Yin. An automatic multi-sample 3d face registration method based on thin plate spline and deformable model. In *IEEE*

International Conference on Multimedia and Expo Workshops (ICMEW), pages 453–458, 2012.

[57] N. Rafibakhsh, J. Gong, M. K. Siddiqui, C. Gordon, and H. F. Lee. Analysis of xbox kinect sensor data for use on construction sites: Depth accuracy and sensor interference assessment. In *Construction Research Congress 2012 sConstruction Challenges in a Flat World*, pages 848–857. ASCE.

[58] Z. Ren, J. Meng, J. Yuan, and Z. Zhang. Robust hand gesture recognition with Kinect sensor. In *Proc. 19th ACM International Conference on Multimedia*, pages 759–760. ACM, 2011.

[59] S. Roberts and L. Stals. Discrete thin plate spline smoothing in 3D. *ANZIAM Journal*, 45:C646–C659, 2004.

[60] ROS.org. Kinect calibration. `www.ros.org/wiki/kinect_calibration/technical`. Accessed: 03/13/2013.

[61] ROS.org. ROS OpenNI. `www.ros.org/wiki/openni_kinect`. Accessed: 03/03/2013.

[62] Y. Schröder, A. Scholz, K. Berger, K. Ruhl, S. Guthe, and M. Magnor. Multiple kinect studies. Technical Report 09-15, ICG, Technical University of Braunschweig, Oct. 2011.

[63] J. Smisek, M. Jancosek, and T. Pajdla. 3D with Kinect. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1154–1160. IEEE, 2011.

[64] E. E. Stone and M. Skubic. Evaluation of an inexpensive depth camera for passive in-home fall risk assessment. In *5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pages 71–77. IEEE, 2011.

[65] W. Susanto, M. Rohrbach, and B. Schiele. 3D object detection with multiple Kinects. In *European Conference on Computer Vision (ECCV) Workshops and Demonstrations*, pages 93–102. Springer, 2012.

[66] T. Svoboda. A software for complete calibration of multicamera systems. In *Electronic Imaging 2005*, pages 115–128. International Society for Optics and Photonics, 2005.

[67] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multicamera self-calibration for virtual environments. *Presence: Teleoperators & Virtual Environments*, 14(4):407–422, 2005.

[68] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[69] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan. Scanning 3D full human bodies using Kinects. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):643–650, 2012.

[70] J. Tran, A. Ufkes, M. Fiala, and A. Ferworn. Low-cost 3D scene reconstruction for response robots in real-time. In *Proc. IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 161–166. IEEE, 2011.

[71] G. Turk and J. F. O'brien. Variational implicit surfaces. 1999.

[72] L. Vera, J. Gimeno, I. Coma, and M. Fernández. Augmented mirror: interactive augmented reality system based on Kinect. In *Human-Computer Interaction–INTERACT 2011*, pages 483–486. Springer, 2011.

[73] G. Wahba. *Spline models for observational data*, volume 59. Society for Industrial Mathematics, 1990.

[74] A. Weiss, D. Hirshberg, and M. J. Black. Home 3D body scans from noisy image and range data. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 1951–1958. IEEE, 2011.

[75] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. Technical Report MIT-CSAIL-TR-2012-020, Massachusetts Institute of Technology, 2012.

[76] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proc. 23nd Annual ACM Symposium on User Interface Software and Technology*, pages 273–282. ACM, 2010.

[77] L. Xia, C.-C. Chen, and J. Aggarwal. Human detection using depth information by kinect. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 15–22. IEEE, 2011.

[78] C. Zhang and Z. Zhang. Calibration between depth and color sensors for commodity depth cameras. In *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2011.

[79] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. 7th IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 666–673. Ieee, 1999.

[80] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[81] Z. Zhang. Microsoft Kinect sensor and its effect. *IEEE Multimedia*, 19(2):4–10, 2012.

[82] M. Zollhöfer, M. Martinek, G. Greiner, M. Stamminger, and J. Süßmuth. Automatic reconstruction of personalized avatars from 3D face scans. *Computer Animation and Virtual Worlds*, 22(2-3):195–202, 2011.