4D foot scanner prototype development

Jure Vidmar

TU Delft

4D foot scanner prototype development

by

Jure Vidmar

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday, December 18th, 2020.

TU Delft

Abstract

Understanding the shape of a human foot is a challenge approached by a wide range of disciplines and industries. Traditional methods of obtaining foot measurements are being replaced with newer 3D scanning technologies, obtaining more accurate and repeatable results.

However, there is still a lack of insight into the dynamic morphology of human foot during motion. Obtaining an insight into the shape of the foot during motion is possible, but not widely accessible, due to expensive and custom equipment used in existing 4D foot scanning prototypes.

This work focuses on developing a 4D foot scanning prototype using commodity hardware, aiming to provide better insight into the dynamic foot morphology and make research in this field more accessible. The scanner is based on RGB-D cameras which offer an affordable access to 4D scanning, using active stereoscopic vision technology. Besides a functional scanner which allows obtaining 4D foot scans at 15 fps, measurement of bandwidth limitations and bottlenecks were performed and synchronization between cameras were assesed.

Contents

Abstract iii

1 Introduction 1
1.1 Why 4D foot scanning? . . . 1
1.2 Problem definition . . . 1
1.3 Proposed approach and thesis goals . . . 2
1.4 Challenges and contributions . . . 2
1.5 Thesis outline . . . 2

2 Related work 3
2.1 4D foot scanners . . . 3
2.2 3D foot scanners . . . 4

3 Prototype design 5
3.1 Concept descrption . . . 5

4 Intel RealSense RGB-D cameras 7
4.1 Why RealSense D435i cameras? . . . 7
4.2 Principle of operation . . . 7
4.3 Camera parameters . . . 9

5 Camera accuracy and parameter exploration 11
5.1 Requirements . . . 11
5.2 Accuracy metrics . . . 11
5.3 Methodology . . . 11
5.4 Depth inhomogeneity measurement . . . 12
5.5 Camera parameter tuning . . . 14

6 Camera calibration 17
6.1 What is camera calibration? . . . 17
6.2 RGB-D camera calibration techniques . . . 17
6.3 Our approach . . . 18

7 Pointcloud alignment 19
7.1 The pointcloud registration problem . . . 19
7.2 Coarse alignment by extrinsic calibration of RGB-D cameras . . . 19
7.3 Our solution . . . 20

8 Prototype implementation 23
8.1 Overview . . . 23
8.2 Data acquisition . . . 26
8.3 Data processing . . . 28
8.4 Limitations of centralized system architecture . . . 29

9 Scanning Results 31
9.1 4D foot scans . . . 31
9.2 Static reconstruction accuracy . . . 33

10 Bandwidth measurements and bottlenecks 35
10.1 Overview . . . 35
10.2 USB theoretical background . . . 35
10.3 Measuring USB data troughput . . . 37
10.4 Discussion . . . 39

11 Camera synchronization **41**
11.1 Overview . . . 41
11.2 Color and depth frame synchronization . . . 42
11.3 Inter-camera synchronization delays . . . 43

12 Modular system architecture: proof of concept **45**
12.1 Overview . . . 45
12.2 Acquisition module: RealSense D435i + Raspberry Pi 4 + SSD . . . 46
12.3 Performace limitations . . . 47
12.4 Synchronization between modules . . . 49

13 Conclusions **51**
13.1 Summary . . . 51
13.2 Future work and potential improvements . . . 51

Bibliography **53**

1 Introduction

1.1. Why 4D foot scanning?

Measurements of human feet are playing an important role in various disciplines ranging from ergonomics and sports to medicine and anthropometry. Footwear design is an example of a discipline which heavily relies on understanding the dimensions and shapes of human feet. Inappropriate shoes can disturb the develpoment of children's feet and lead to deformities [2], while therapeutic footwear can be used for preventing certain foot related diseases [10].

Despite their importance for our well-being, the footwear design processes are based on rather crude measurements and subjective shape understanding of our feet. Most of the footwear is designed based on footprints, obtained using foam, wax or plaster casting and not taking into account the vertical dimensions of the foot. Due to the complex structure of the foot and lack of instrumentation, the vertical dimensions are classified into discrete types by visual inspection and based on plantar dimensions [32].

Some of those limitations have already been addressed with 3D reconstruction technology and advancements in computer vision research. Three dimensional foot scanners are now being used not only for research, but already commercially available and might soon become a part of our everyday lives [36]. However, there is still a lack of 4D information regarding foot morphology during motion. Size of certian dimensions can change up to 8mm between load-bearing and non-weight-bearing states [39] and midfoot girth changes significantly during walking which is often not taken into account while designing shoes [6]. Four-dimensional foot scanners can provide insight into these dynamic properties of human feet.

Several attempts at 4D foot scanning have been performed, but their implementation is still very complex and not easily reproducible. Furthermore, most of the 3D scanners reaching sub-millimeter accuracy are not fast enough for dynamic capture of the foot in motion. While there are several existing 4D foot scanners, their cost and complexity renders them unsuitable for clinical and commercial use [36]. The appearance of commercial RGB-D cameras offers a low-cost approach to 3D scanning and might enable widespread usage of dynamic foot scanning applications in near future.

1.2. Problem definition

Currently, there are no 4D foot scanners available commercially and existing research prototypes are expensive and hard to reproduce. These prototypes use expensive and custom-made equipment, which makes them unsuitable for clinical and commercial use. Creating an affordable 4D foot scanning device from commodity hardware would provide better insight into the morphology of the human foot and open new possibilities in medicine, sports and ergonomics.

1.3. Proposed approach and thesis goals

The goal of this graduation project is to develop a functional 4D foot scanner prototype, which would offer an insight into the shape of human foot during motion. The problem of a prototype development can be expanded to the following objectives:

- Literature study of existing foot measurement techniques, focusing on 3D and 4D scanning technologies.
- Evaluating the accuracy and suitability of commercial RGB-D cameras 4D foot scanning applications
- Developing a functional 4D foot scanning prototype using multiple RGB-D cameras, resulting in a sequence of three-dimensional frames (pointclouds) of the foot during walking.

The goal of this graduation project is to estabilish a starting point for dynamic foot morphology research, by creating a 4D foot scanning prototype, which will enable future students and researchers to work upon.

1.4. Challenges and contributions

The development of a 4D foot scanner poses many challenges due to its interdisciplinary nature, encompassing computer vision, embedded systems, electronics, mechanical design and ergonomics.

Despite being a heavily researched topic in computer vision, multi-camera 4D reconstruction is still not a trivial task. Due to inacurracy of commodity RGB-D cameras (compared to industrial scanners), the camera calibration and parameter settings play an important role in 4D reconstruction quality. In scenarios with multiple cameras, the pointcloud alignment challenge needs to be solved. On the other hand, the high bandwidth requirements of commercial RGB-D cameras and their timing synchronization pose a significant embedded system design challenge, specially when using affordable, off-the-shelf equipment. Lastly, there are many unforeseen problems of practical nature which rarely make it to scientific literature, yet often limit the scope of research experiments and their success.

The main contribution of this graduation project is a working 4D foot scanning prototype, built from commodity hardware. It uses four RGB-D cameras, capturing color and depth data at 15 fps with resolution of 848 x 480 pixels. Besides a functional prototype, an alternative system architecture was investigated in order to address the limitations which were observed during the evaluation process. The concept of a modular scanner architecture is demonstrated using Raspberry Pi computers, focusing on data acquisition aspects.

1.5. Thesis outline

Related work in the field of 4D and 3D foot scanning is presented in Chapter 2, followed by a brief description of the foot scanner concept in Chapter 3. The RealSense RGB-D cameras used in our work are presented in Chapter 4 and the accuracy and parameters of the D435i camera model are discussed in Chapter 5. The calibration of cameras is discussed in Chapter 6, followed by the pointcloud alignment challenge in Chapter 7.

The resulting prototype implementation is presented in Chapter 8 and scanning results are discussed in Chapter 9. In Chapter 10 we investigate bandwidth requirements and bottlenecks, focusing on USB data transport. Camera synchronization measurements are shown in Chapter 11. An alternative system architecture addressing the limitations of our prototype is presented in Chapter 12, followed by conclusions in Chapter 13.

2

Related work

2.1. 4D foot scanners

The need for an insight into dynamic properties of a human foot during motion is present already for years and there have been several attempts of addressing it.

One of the earliest dynamic 3D foot reconstruction systems was designed by Cordet et al. [14], using 3 pairs of custom stereoscopic sensors made out of industrial-grade IEEE1394 cameras. Although this technique provided dynamic foot reconstruction at 49 FPS and resolution of 640 x 480 pixels, it involved adding an extra factor to the foot surface (either socks or paint), something that is undesirable in a clinical environment [38].

Schmeltzpfenning et al. [31] developed a system for scanning foot in motion using custom-made scanner units based on a CCD camera and a projector. It allowed data acquisition at 41 FPS with a resolution of 640 x 480 pixels.

Several years later, Kimura et al. [26] developed a system for measuring feature cross-sections of a foot in motion, but did not reconstruct the complete 3D shape and required manually marking features on the foot's surface. The proposed system used 12 industrial-grade cameras operating at 1024 x 768 resolution at framerate of approximately 14 fps.

Liu et al. [1] designed a dynamic foot scanning system using three time-of-flight cameras, operating at resolution of 176 x 144 pixels and frequency of 50 fps.

Another approach using multiple-laser-plane triangulation was presented by Novak et al. using four camera pairs and a laser projector [30]. It captured complete foot during walking at 30 fps framerate and resolution of 640 x 480 pixels.

Thabet et al. [38], presented an overview of existing dynamic 3D foot scanning approaches and developed a prototype using a single camera and a coded-light projector. The plantar surface of the foot was obtained at 60 fps and HD resolution of 1920 × 1080 pixels. Unlike many other reports which focused mostly on technical aspects of their prototypes, Thabet et al. included accuracy and repeatability assesment with real feet scans, reaching average accuracy of 2.8 mm in dynamic case.

A dynamic foot scanner was build by a company called 3dMD, which creates scanning modules using custom-made stereoscopic cameras and a speckle pattern projector [4]. According to the manufacturer, it allows up to 120 fps acquisition framerate reaching reconstruction accuracy of 0.7 mm.

More recently, Boppana et al. [8] developed a dynamic 3D scanner using RealSense D415 cameras and used it for statistical shape modelling of human feet [9]. Six RealSense RGB-D cameras were connected to a single computer, streaming at 90 fps. However, the resolution used and foot reconstruction accuracy were not specified.

Most of the foot scanners mentioned above are custom-built devices, which require expensive equipment and technical expertise. Multiple scanners provide only partial measurements of the foot, such as the plantar or dorsal surface, while obtaining sufficient, sub-millimeter 4D reconstruction of complete foot's surface using commodity hardware is still being researched.

2.2. 3D foot scanners

Static three-dimensional reconstruction of human foot is more developed compared to dynamic scanning. There are numerous approaches of obtaining 3D shape of a foot, but most of technologies used are not suitable for dynamic scanning due to their slow acquisition rate.

A paper by Telfer et al. [36] provides an insight into different methods of obtaining 3D foot scans and their usages for foot research. Already at that time (2010) there were not only research prototypes, but commercially available solutions in the price range between €5,000 and €30,000 [36].

Wan et al. [43] created a scanning device using five custom built active steroscopic scanning modules, based on Cannon Powershot A650IS and pattern projectors. The proposed system was evaluated against real feet, allowing sub-millimeter (0.8mm) reconstruction accuracy.

Ballester et al. [5] compared the validity and reliability of different data-driven 3D scanning techniques, where one of them was 3D foot reconstruction using nothing more but a smartphone application. Accuracy in terms of 3D reconstruction was not assessd.

More recently, several foot scanners were designed using consumer RGB-D cameras. Rogati et al. performed validation of the Kinect cameras, reconstructing the shape of a foot with approximately 2.8 mm accuracy. Similar technology was used in a prototype by Yuan et al.[45], which was powered by five Intel RealSense SR3000 cameras. The accuracy in terms of 3D reconstruction was not assessed.

One of the most recent and radically different approaches to measuring static shape of a human foot was presented by Zhang et al. [47]. Unlike previous scanners which were based on optical technologies, Sensock uses wearable-electronic sensors for obtaining the 3D shape. A sock made out of conductive materials measures feet's dimensions based on resistance changes due to streching of the material.

Besides above-mentioned foot scanners, there are many commercially available scanners used in research. Witana et al. used the YETI scanner [44] and Stankovic et al. [33] used the Tiger 3D laser scanner by RS Scan, for example.

3

Prototype design

3.1. Concept descrption

One of the main goals of this graduation project is designing and building a 4D foot scanner prototype. In order to scan a foot during motion, there is an elevated stage with a transparent section (plexi glass) which allows capturing a foot from all sides including its bottom. Subject walks across the stage and a single step is captured and processed by the scanner. Sketch of the prototype concept is shown in Figure 3.1.

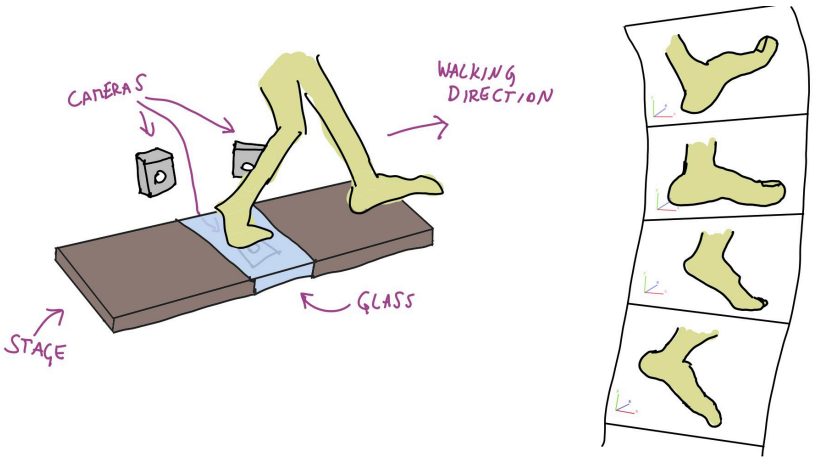

Figure 3.1: Sketch of the foot scanner prototype concept. RGB-D cameras placed around the transparend section of a walking stage allow capturing the foot from all sides. Resulting output is 4D reconstruction of the foot during walking (capturing a single step).

The desired output is a 4D reconstruction of a foot in motion: a sequence of three dimensional frames, called pointclouds. This kind of 4D data can be used for measuring foot's dimensions in footwear industry, orthotics and prostethics design and research of the dynamic foot morphology.

Commercially available RGB-D cameras are evaluated and used in our prototype, due to their compact form, affordability and widespread adoption in the research community. Multiple RGB-D cameras synchronously capture data, which is processed and visualized offline. Since our goal is to design an easily reproducible system using commodity hardware, the data acquisition and data processing steps can be decoupled, favouring accuracy over real-time performance.

We approach the prototype development, by first investigating the accuracy and functioning of an RGB-D camera, the RealSense D435i. Second, the 4D reconstruction aspects of the prototype are researched, solving the problems of camera calibration and pointcloud alignment. Finally, a physical scanner prototype is built in order to verify the selected approach and implementation.

Upon obtaining a successful 4D foot scan, the system is evaluated in terms of scanning accuracy, data acquisition capabilities and scalability. Observed limitations are addressed in the second iteration

of prototype design, where modular system architecture is explored and tested using single-board computers.

4

Intel RealSense RGB-D cameras

4.1. Why RealSense D435i cameras?

The appearance of commercially available RGB-D cameras on the marked gained a lot of attention in the research community. Custom-made depth and color sensing devices were built and used in research before, while the convenience and affordability of consumer RGB-D devices allowed researchers to focus on applicational aspects, without requiring deep expertise in computer vision.

At the time of writing, Intel RealSense D415 and D435 were the newest RGB-D camera models promising good depth sensing capabilities. Choosing between the two, the D435 is better suitable in our scenario due to its wide angle of view, allowing for near placement and lower number of cameras needed. Furthermore, the global shutter of D435i captures all pixels simultaneously (unlike the rolling shutter used on D415 which scans through pixels sequentially) which allows capturing dynamic scenes without artefacts [13].

4.2. Principle of operation

Intel RealSense D435i camera is an active stereoscopic depth camera, which contains three image sensors and an infrared projector, shown in image 4.1.

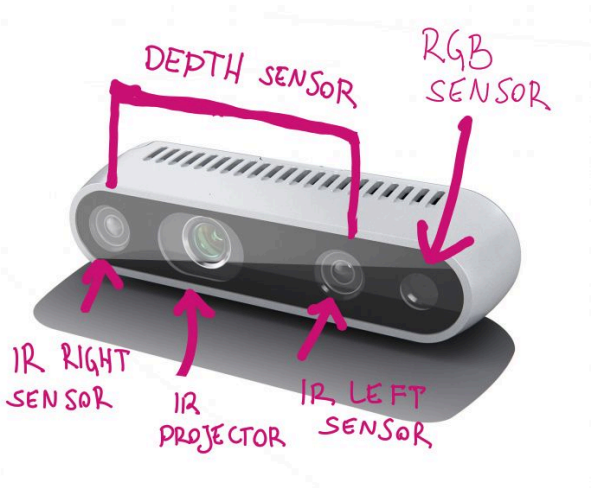

Figure 4.1: Intel RealSense D435i camera. Depth sensor consists of a stereoscopic pair of near-inrared sensors (IR left and IR right) and a infrared pattern projector. Color sensor (RGB sensor) provides regular color images. Original image source: https://newsroom.intel.com/news/new-intel-realsense-d435i-stereo-depth-camera-adds-6-degrees-freedom-tracking/

A pair of image sensors is used for generating depth frames, using stereoscopic vision techniques while the third image sensor provides regular color frames. The infrared projector is used to improve depth accuracy in scenes with low texture, for example when facing a white wall. The lack of visible

features can be compensated for, by projecting a pattern of dots onto the wall (called *active* stereoscopic depth sensing) providing artificial feature points which are used for trianglation in the process of depth calculation [24].

From the user's perspective, the RealSense cameras can be seen as a black-boxes providing multiple kinds of streams. The RGB image sensor outputs color frame stream and the IR sensors output IR frame streams. Since the stereo-to-depth calculation is performed in camera's internal vision processor, there is also a depth stream provided by a virtual depth sensor. Additionally, the D435i model contains has onboard IMU (Intertial Measurement Unit) which is not used in our application.

Combination of a depth and color image, called a RGB-D image, can be converted into a three-dimensional pointcloud as shown in image 4.2. Conversion from depth images to 3D point clouds is trivial, assuming that intrinsic parameters of the camera (described in chapter 6) are known.

Figure 4.2: Color image, depth image and a textured pointcloud

Every pixel of the depth image represents the z-distance between the camera and the scene (e.g. a 16-bit integer value, representing depth in millimeters). Unlike range images whose pixel values are designating distance between camera's origin and observed world points, the depth image takes into account only the depth (z) component of this distance, as shown in image 4.3.

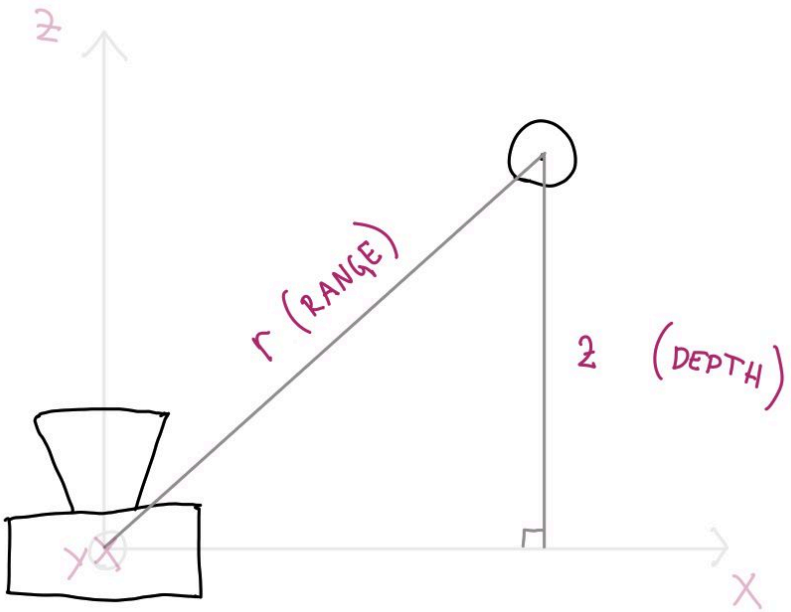

Figure 4.3: Comparison between depth and range measurements. A single pixel of a depth image contains a single depth value.

4.3. Camera parameters

The RealSense D435i camera is a complex device with multiple tunable parameters affecting the quality of RGB-D data.

The *resolution*, *framerate* and *exposure time* can be changed for both, color and depth sensors, described in previous chapter 4.2. Increasing resolution (number of pixels in a frame) allows more detailed color and depth images, while the framerate (frames per second) affects the quality of RGB-D data in dynamic scenes. All parameters mention above affect the bandwidth requirements of the device, posing a tradeoff between the quality of RGB-D data and its bandwidth requirements.

Depth sensor outputs data in an unsigned 16-bit unitless format, quantizing measured distance into discrete steps. This step size is determined by the *depth unit* parameter, with the smallest possible value of 100 µm which allows the maximum sensing range of approximately 65 m (limited by 16-bit unsigned integer). This can be increased for larger operating ranges with the price of larger quantization errors.

The near-infrared pattern projection power is controlled with the *laser power* (in mW) parameter. Increasing its value can enhance the SNR of the projected pattern at long distances, but might degrade depth sensing performance in close ranges due to laser speckle.

There are more parameters related to details of stereo-to-depth calculation available, which we do not investigate in our work.

5

Camera accuracy and parameter exploration

5.1. Requirements

Commercial RGB-D cameras are known for their distortions and significant depth measurement errors, compared to professional 3D scanners. Until recently, most of the antrophometric databases used by designers and manufacturers contained only 1D and 2D data with manually obtained measurements [37]. The use of 3D scanning and digitizing technologies improved on the accuracy and repeatability of those measurements, while sub-millimetre accuracy is nonetheless sufficient for the purposes of footwear design [25]. For the purposes of our project, we aim to achieve sub-millimetre 4D reconstruction accuracy of the human foot during walking.

5.2. Accuracy metrics

In order to obtain insight into the capabilities of the RealSense D435i camera we performed several accuracy measurements, inspired by a paper [11] conducting an in depth metrological characterization of a similar camera, RealSense D415, assesing different kinds of errors at different operating ranges (distances from the camera).

There are two kinds of systematic errors which are characterizing the camera's performance:

- **Depth offset** (or depth-accuracy, distance-accuracy, absolute error) is the difference between the measured depth values and actual depth values.

- **Depth inhomogenity** (or spatial noise, non-uniformity, planarity) is the variation of depth values across pixels, when measuring a planar target (e.g. a flat wall).

Monica Carfagni et al. evaluated the camera not only in terms of depth-offset and depth-inhomogeneity, but performed multiple experiments using special artefacts prescribed by the German standard VDI/VDE 2634 Part 2 ("Optical 3D measuring systems") and additional experiments which are specific for characterizing depth cameras, such as 3D reconstruction accuracy [11]. Due to the large scope of our project and limited accessibility to professional measuring equipment (COVID-19 pandemic lockdown) we only assessed the camera's depth inhomogeneity.

5.3. Methodology

Depth inhomogeneity is evaluated by facing the camera perpendicularly torwards a white wall and capturing depth images at distances between 200 mm and 800 mm with a pitch of 100 mm. Root-mean squared error (RMSE), shown in formula 5.1, is calculated using signed distances between the scanned data and best fitting plane built on top of this scanned data. Ideal depth camera measuring a perfectly flat plane would have RMSE of 0 mm.

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N} signed\_distance^2}{N}} \tag{5.1}$$

Since misaligning the camera would bias the results, a best-fit plane is first computed on the captured data and RMSE is measured using distances between measured points and the best-fitted plane, rather than relying on ground truth. Plane fitting is performed using linear regression[1], minimizing the sum of squared distances between the plane and data points. Formula for calculating the distance between the best-fit plane ($Ax + By + Cz = D$) and a point (x, y, z) is shown below 5.2.

$$signed\_distance = \frac{Ax + By + Cz - D}{\sqrt{A^2 + B^2 + C^2}} \tag{5.2}$$

5.4. Depth inhomogeneity measurement

Using the methodology described in previous chapter, we positioned the RealSense camera against a wall and measured the RMSE between obtained points and best fitting plane. Experimentation setup is shown below.

The camera was set to default settings, with depth resolution of 848 x 480 pixels, 6500 µs exposure time and laser power of 150 mW. The only parameter changed was the depth unit, which was set to 0.0001 m in order to achieve best depth resolution. Results of the experiment are shown in figure 5.1, where each datapoint represents an average RMSE calculated from 50 frames.

[1] Ordinary least squares linear regression from SciKit Learn (python machine learning library) was used in our experiments.

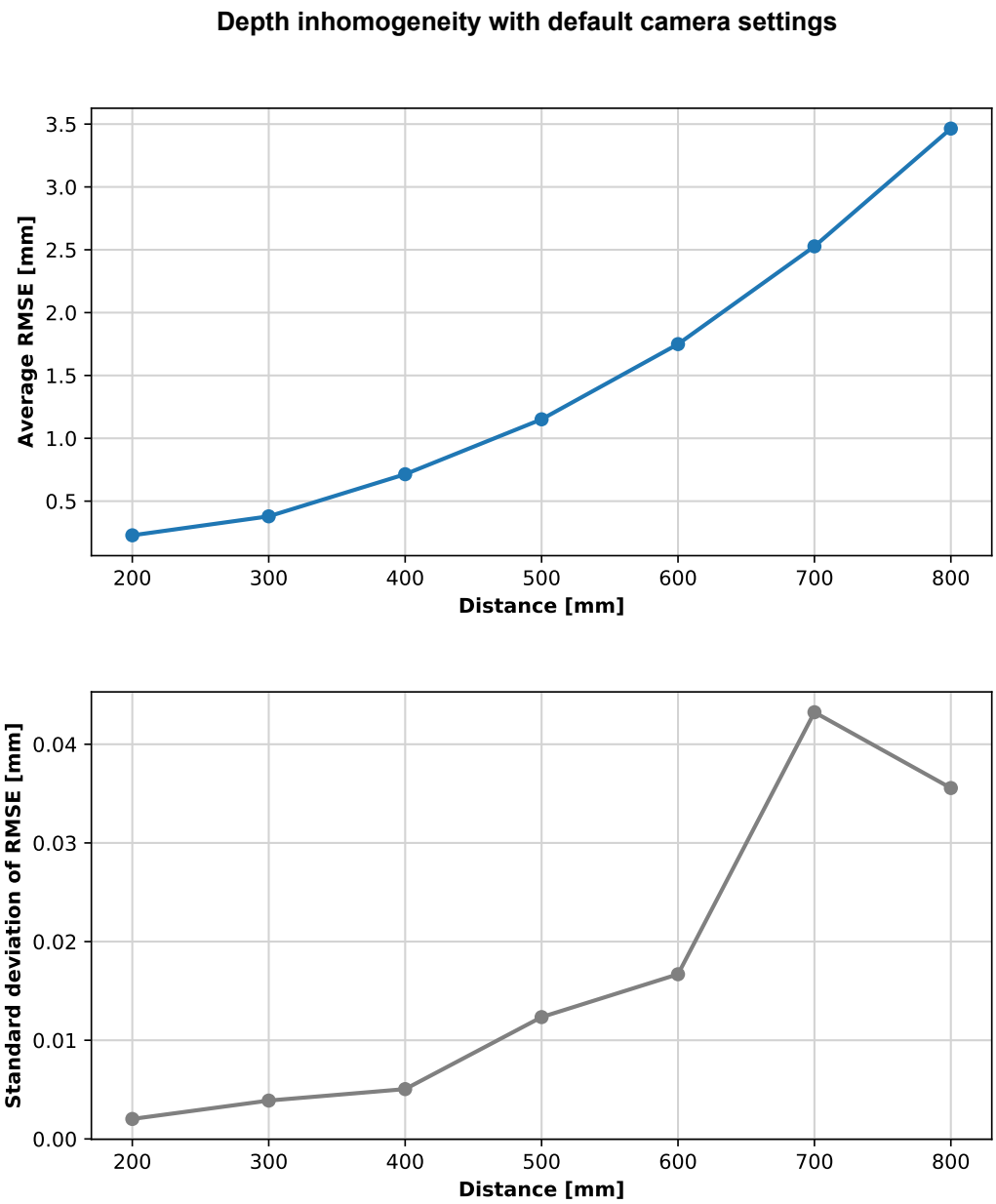

Figure 5.1: RMSE at distances between 200 mm and 800 mm, using default camera settings: resolution 848 x 480 px, exposure 6500 µs, laser power 150 mW and depth unit of 0.0001 m

5.5. Camera parameter tuning

In order to explore how changing parameters such as resolution, exposure time and laser power affects the depth measurement quality, the depth inhomogeneity measurements from previous chapter 5.4 were repeated with modified camera settings.

Resolution for both color and depth sensors was evaluated first (figure 5.2), from lowest (480x270 px) to highest (1280x720 px), at same distances as in the previous experiment. In general, higher resolution should provide greater accuracy, but our measurements indicate 848x480 px resolution as optimal which confirms statement in official RealSense D435i documentation [24].

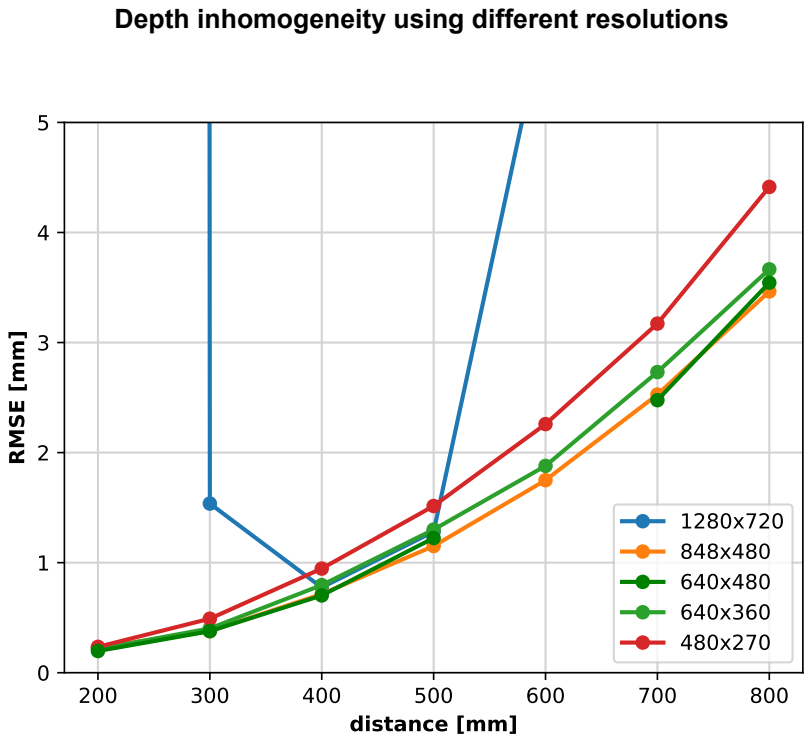

Figure 5.2: Comparison of different resolutions (color and depth sensor) at different distances, measuring depth inhomogeneity in millimeters. Other camera settings are set to defaults.

Laser power and exposure time were evaluated together, since short exposure times with high laser powers might increase SNR, specially in circumstances where sunlight or other sources of near-infrared light are interfering with the projection. Results at distances 200 mm, 500 mm and 800 mm can be observed in the graph below 5.3. Exposure time and laser power need to be increased for better results at larger distances.

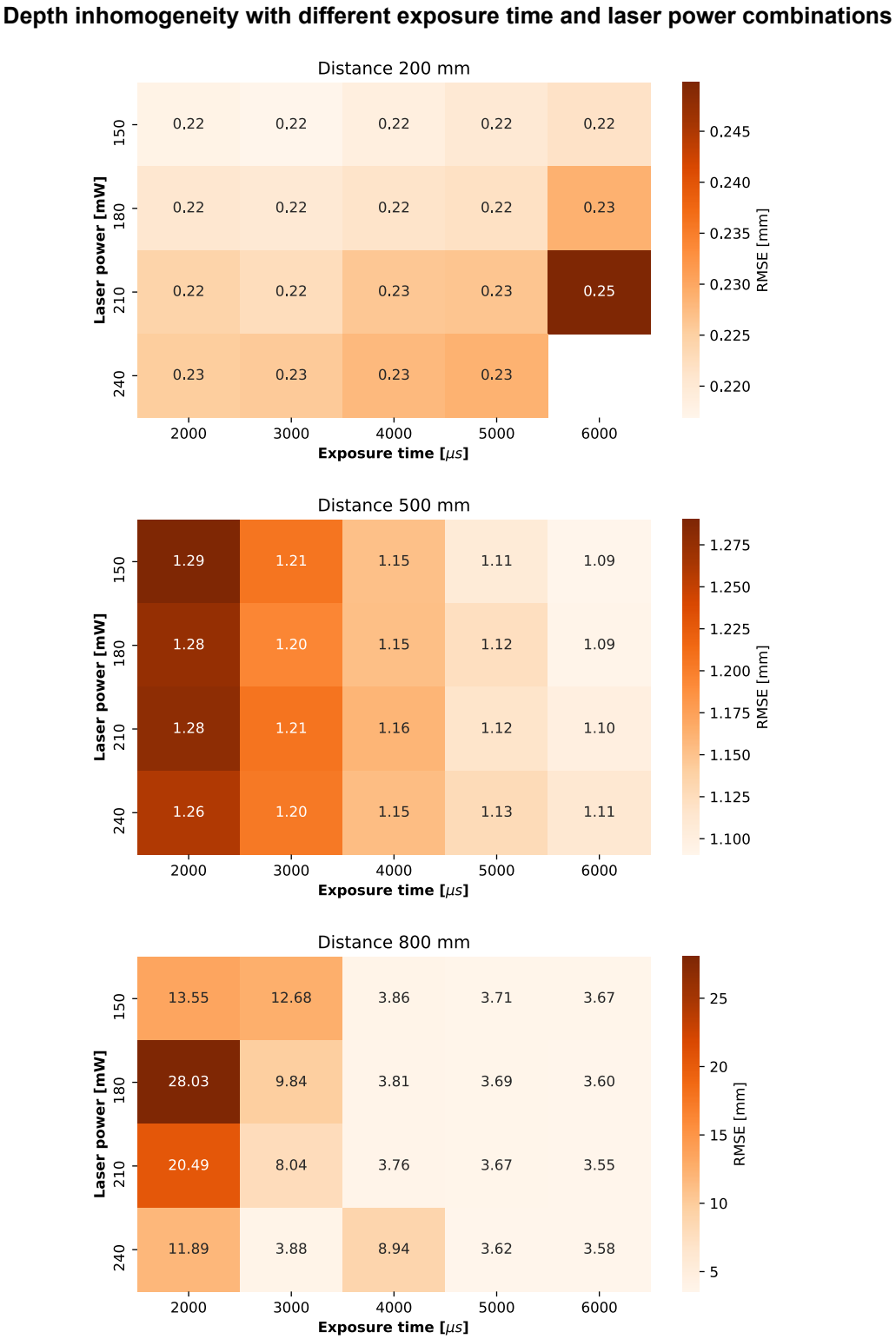

Figure 5.3: Comparison of different exposure and laser power settings at distances 200 mm, 500 mm and 800 mm. Measurement at distance 200 mm (240 mW, 6000 µs) was removed for beter graph readability (outlier).

6

Camera calibration

6.1. What is camera calibration?

The accuracy evaluation of the RealSense D435i in previous chapter was performed using its default settings in order to evaluate the camera in a simple and repeatable manner. However, stereoscopic depth measurements are heavily dependent on internal camera parameters which are often poorly estimated by manufacturers and can be improved by re-calibrating the cameras [28].

Camera calibration is the process of estimating the parameters of the pinhole camera model, most widely used mathematical representation of a camera. It describes the internal properties of the camera such as focal length, sensor size and pixel size (intrinsics) and its pose in world's coordinate system (extrinsics). These parameters can be represented in a 3x4 camera matrix C, which describes the projection of 3D world points to the 2D image plane, shown in formulas 6.1, 6.2.

$$x_{image} = C * X_{world} \tag{6.1}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p1 & p2 & p3 & p4 \\ p5 & p6 & p7 & p8 \\ p9 & p10 & p11 & p12 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{6.2}$$

One of the most widely used calibration approaches is Zhang's technique [48], which uses a known planar target (e.g. a chessboard) for finding correspondences between 3D world points and 2D image points. The positions of chessboard corners[1] in 3D space are known and assumed to be coplanar with Z coordinate equal to zero. Image of a chessboard is then used for detecting 2D corner positions, estabilishing an over-determined system of equations. Solving this equational system results in an estimation of the camera matrix, describing camera's intrinsic and extrinsic parameters. Despite not being included in the pinhole camera model due to non-linearity, the effects of lens distortion are also estimated during camera calibration process and might significantly affect the quality of stereoscopic 3D reconstruction.

There have been many derivations and improvements on the original technique, using different calibration targets, feature detection algorithms and optimization procedures. However, those are not investigated further due to the scope of our project and widespread usage of the chessboard calibration technique.

6.2. RGB-D camera calibration techniques

Most of the optical methods of calibration can be applied to RGB-D cameras, by using one of the IR imagers or a color imager. However, there have been many proposals for calibration techniques specific to RGB-D cameras improving on calibration accuracy:

Staranowicz et al. developed a RGB-D calibration technique using a spherical object (e.g. a basketball), which provides results comparable to chessboard based techniques and is easier to use [34].

[1] Intersections between internal chessboard squares

A popular technique for calibrating RGB-D cameras was introduced by Herrera et al, using a chessboard for calibrating both RGB and depth sensors simultaneously [23]. Planar calibration target is detected on both color and depth images and accurate estimation of both sensor's parameters are estimated using non-linear bundle adjustment techniques.

Beck et al. calibrated multiple RGB-D cameras by attaching tracking markers to the chessboard and capturing it within a motion capture system (often used in animation and film industry) which improved on the accuracy of chessboard pose detection [3]

An overview of RGB-D specific calibration techniques was presented by Villena-Martínez et al. [42].

6.3. Our approach

Despite being a heavily researched topic in computer vision community, camera calibration is still a complex and often tedious process where obtaining good results is not trivial. In the case of a stereoscopic RGB-D camera, calibration complexity further increases as there are three image sensors that need to be calibrated.

While each of the sensors could be calibrated using the same principle, with a chessboard, we use manufacturer's software for calibrating the camera. Intel RealSense provides the Dynamic Calibrator [12] which allows calibrating the extrinsics of all sensors, using a custom calibration pattern printed on paper or displayed on a smartphone screen, shown in image 6.1. Unfortunately, the software does not calibrate intrinsic parameters and lens distortion coefficients, which are required during pointcloud alignment phase (7). In order to obtain the missing parameters, we perform additional chessboard calibration on the RGB sensor using OpenCV calibration functions.

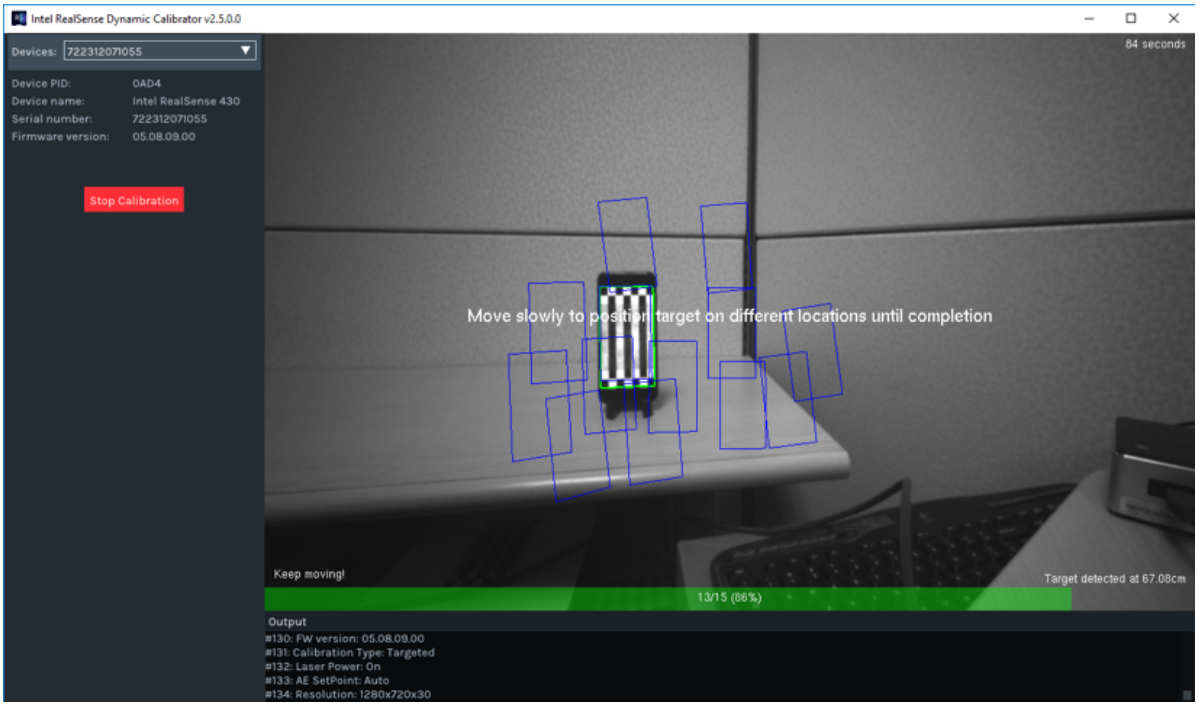

Figure 6.1: Intel RealSense software for camera calibration, Dynamic Calibrator. Image source: dev.intelrealsense.com

7

Pointcloud alignment

7.1. The pointcloud registration problem

Reconstructing the 3D shape of a foot using multiple cameras requires aligning the captured pointclouds into one coordinate space. Pointcloud alignment or registration is one of the fundamental computer vision topics which seeks the transformation that best aligns two pointclouds together [20].

Various registration methods can be classified into coarse and fine methods, where coarse algorithms provide rough initial alignment, which can be refined using fine registration algorithms [35]. The latter are often computationally intensive and of iterative nature, which is why coarse registration is often used in practice.

Most of the registration algorithms operate solely on 3D data, minimizing the distance between two pointclouds, without any knowledge about the data acquisition setup. However, information about camera positions in space can be obtained during calibration phase and used for coarse alignment of pointclouds. This kind of pointcloud fusion is often used in virtual reality and telepresence setups using multiple RGB-D cameras, since the fine registration algorithms are computationally too expensive for real-time performance.

7.2. Coarse alignment by extrinsic calibration of RGB-D cameras

Camera calibration process described in chapter 6 provides information about the intrinsics and extrinsics of the camera, where the latter contains rigid transformation (rotation and translation) between the calibration target and the camera, as shown in image 7.1. Observing a calibration target with multiple cameras simultaneously allows determining their positions in a single coordinate space.

In case of RGB-D cameras, there are multiple sensors which could designate the camera's origin: pointclouds are measured from the depth origin of the camera, which is located at the left IR sensor, while the color images used in calibration process originate from the RGB sensor. This can be compensated for, by registering depth to color or vice versa, using the extrinsic transformation matrix obtained during camera calibration (Chapter 6). With depth and color frames of an RGB-D camera aligned, chessboard and other optical calibration techniques can be directly applied.

Aligning pointclouds by means of extrinsic camera calibration with a chessboard has been used before: Kimura et al. calibrated multiple cameras using a chessboard cube, rather than a plane, for reconstructing the shape of a foot in motion [26]. Maimone et al. used a pairwise chessboard calibration approach with Kinect RGB-D cameras for real-time telepresence setup [29]. Beck et al. registered multiple RGB-D cameras into a common coordinate system by attaching tracking markers to the chessboard and capturing it within a motion capture system (often used in animation and film industry) [7]. Since their original approach was lengthy due to predefined static locations of the calibration target, they refined the process, allowing random positioning of the calibration target [3].

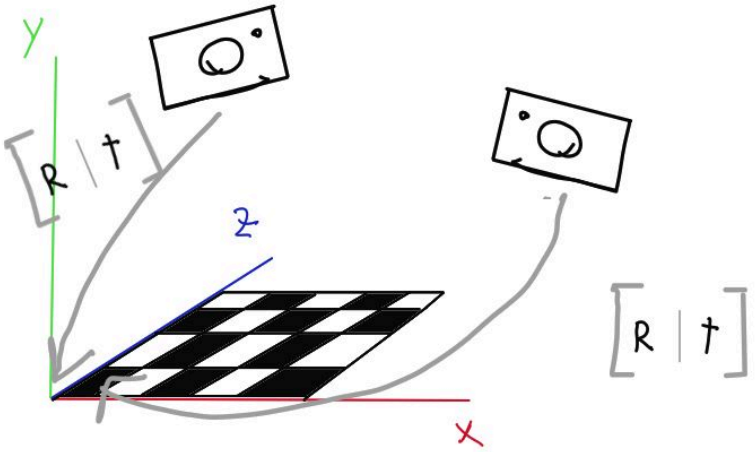

Figure 7.1: Calibration of two cameras using a chessboard. Resulting transformation matrices (extrinsics) can be used for determining position of cameras in relation to the chessboard.

7.3. Our solution

Our approach to pointcloud alignment relies on extrinsic camera calibration, as described in previous section 7.2. It is impossible to capture the calibration target with all cameras simultaneously, which is why we perform pairwise calibration followed by calculation of relative transformations between camera pairs, similar to [29].

A chain of subsequent camera pairs is determined manually and for every pair, multiple images of the calibration target in various positions and orientations are taken [1]. The first camera in the chain is used as a world origin, on top of which other cameras are registered, as shown in image 7.2. Every view of the calibration target observed by a camera pair results in both camera's poses (extrinsics), which are used for calculating the relative transformation between them. Resulting transformation matrices are averaged, obtaining a 4x4 homogeneous matrix describing the rotation and translation from camera A to camera B within a single camera pair.

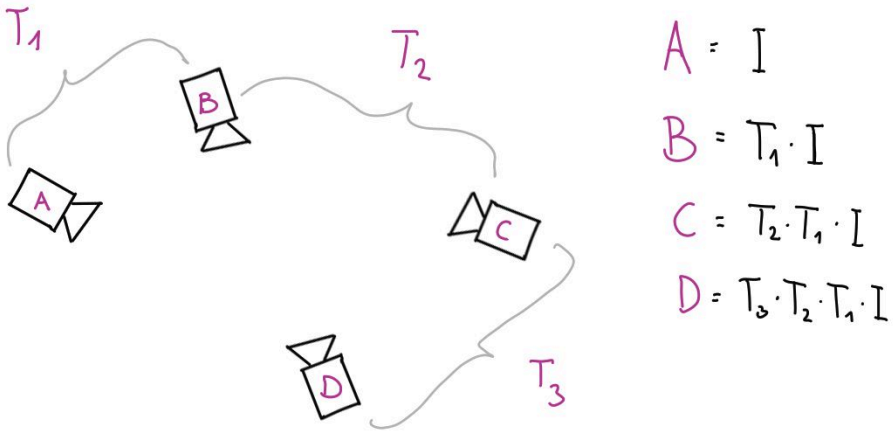

Figure 7.2: Pairwise registration of cameras to a single coordinate system. For every camera pair, a relative transformation is calculated based on their extrinsic calibration (T1, T2, T3). Equations on the right side represent each individual camera's transformation matrix, used for aligning pointclouds to a common coordinate frame (first camera's origin)

Calibrating cameras in a pairwise manner is prone to error accumulation, but allows easier calibration procedure in practice. Furthermore, the chessboard calibration technique does not work on images with only partial view of the chessboard. To overcome this limitation while calibrating cameras in our

scenario, we use ChArUco board instead of chessboard, shown in image 7.3. Calibration procedure is essentially the same, but this time partial views of calibration target are supported as every square within the chessboard carries an unique identifier (ArUco).

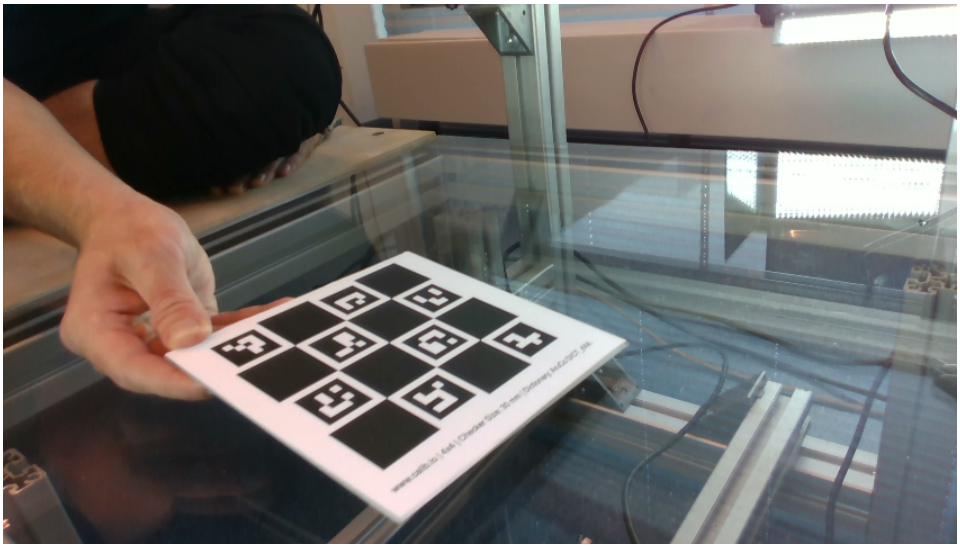

Figure 7.3: Camera calibration using charuco target

[1]Approximately 60 frames per camera pair

8

Prototype implementation

8.1. Overview

The 4D foot scanner concept from chapter 3 was embodied using four RealSense D435i cameras mounted around a transparent section of a walking stage, shown in Figure 8.1.

Cameras are connected to a single computer, used for both data acquisition and processing. It consists of an Intel i7-920 processor, a StarTech USB 3.1 Gen 1 PCI expansion card[1] and a Samsung EVO 850 SSD disk. Computing platform is shown in Figure 8.2.

The stage was built using T-slotted aluminum extrusions (50 mm x 50 mm), wooden panels (18 mm thick multiplex) and plexi glass (15 mm thick extruded acryllic). Cameras are mounted on a separate construction with ballhead mounts and a custom-printed 3D adapter, shown in Figure 8.3.

The implementation of data acquisition and processing is presented in sections 8.2 and 8.3 respectively. Limitations of presented system architecture are discussed in section 8.4

[1] The mentioned USB PCIe expansion card has four USB ports on two separate USB buses (Host Controllers). More details about the USB bus are presented in chapter 10.2.

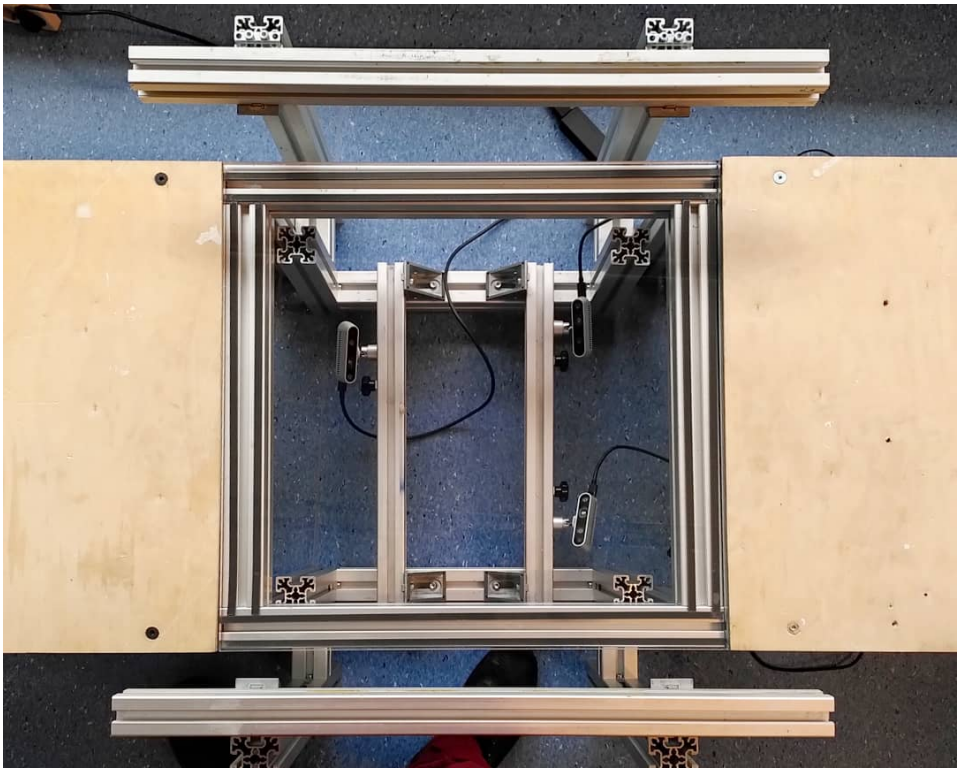

Figure 8.1: 4D foot scanning prototype.

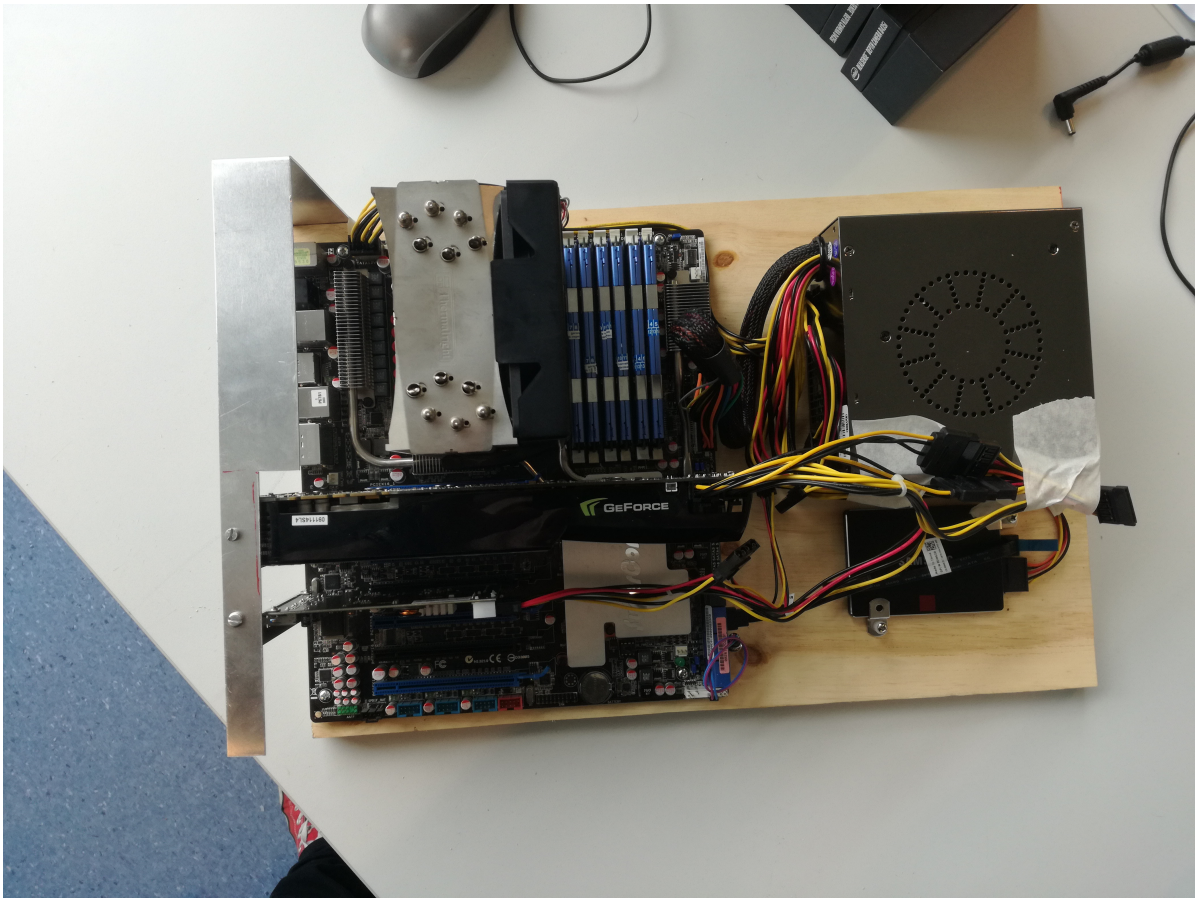

Figure 8.2: Computer used for data acquisition and processing.

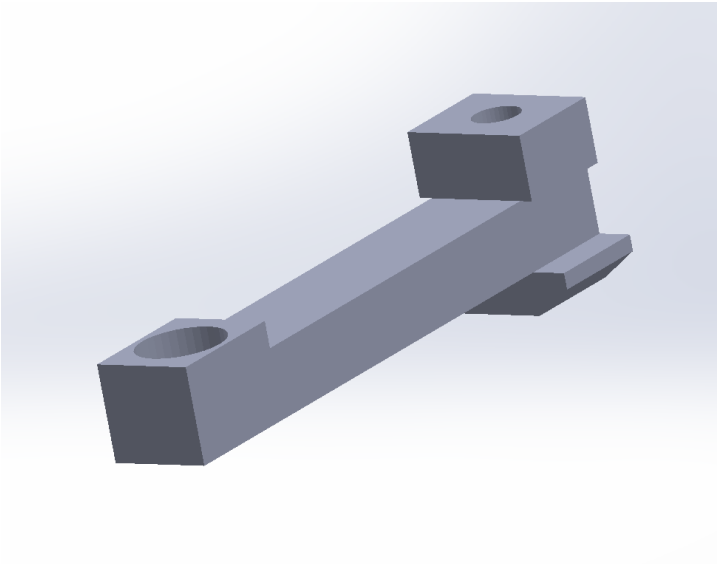

Figure 8.3: The model of an adapter for attaching RealSense cameras to extruded aluminum frame.

8.2. Data acquisition

Data acquisition software is responsible for synchronous streaming of color and depth frames from multiple RealSense cameras and writing them to an SSD drive.

The software is implemented in C++ programming language, using the RealSense SDK which provides an application programming interface (API) to the cameras, abstracting details such as device detection, inter-sensor frame synchronization and registration, metadata handling etc.

The software used for capturing RGB-D data from cameras first initializes the device (lines 1-4), by enabling its color and depth streams, both at resolution of 848 x 480 pixels and framerate of 15 fps[2]. According to our accuracy experiment results (Chapter 5) we disable autoexposure on all sensors, set the projection laser power to 180 mW and configure depth sensor to the smallest depth unit (100 µm). Pseudocode of the program is shown in Algorithm 1.

Algorithm 1 Software used for RGB-D data acquisition (pseudocode)

```
1: Find connected device
2: Enable color stream
3: Enable depth stream
4: Set depth unit, exposure time, laserpower for both streams
5: while streaming do
6:     wait for color+depth frame pair
7:     write color frame to a file
8:     write depth frame to a file
9: end while
```

The users interact with the separate, main program, which handles input parameters (framerate, output destination directory etc.), detects connected devices and spawns a separate instance of the program (Algorithm 1) for every connected camera. Connecting four cameras, for example, creates four separate processes, each capturing RGB-D data from a single camera.

Synchronous data acquisition is achieved by simultaneous start of the RGB-D data streams on all devices. This is achieved by toggling the *streaming* flag on line 5, which is controlled by the main program and is implemented by means inter-process communication primitives (our implementation uses a POSIX semaphore for synchronization between processes). Synchronization delays between cameras are measured and discussed in Chapter 11.

One of the design choices was choosing which kind of data to store on disk: raw color and depth frames or textured pointclouds. Since our application operates on 3D data, there is no explicit need for storing raw frames on disk, when they could be processed into a pointcloud and saved to disk as shown in image 8.4. However, observing the file size of both options showed that there is no benefit in converting data to pointclouds before writing to disk (pointcloud data contains more information resulting in larger flie sizes). As a result, raw RGB-D frames are stored on disk and converted to 3D pointclouds in the post processing stage (Chapter 8.3).

[2]The limitations of the maximum framerate are discussed in Chapter 10.

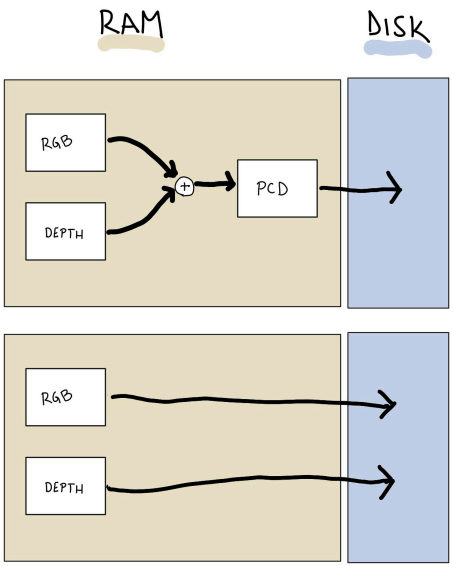

Figure 8.4: Comparison between saving raw color and depth frames (bottom) or converting them to pointclouds first (top). The former option is used implemented, since the size of pointcloud data is larger than size of raw frames.

8.3. Data processing

Upon a successful data acquisition stage, frames from all cameras are processed in order to obtain 4D reconstruction of a foot in motion. This involves converting color and depth frames into textured pointclouds, aligning them into a single coordinate space (using calibration data, described in chapter 7) and storing the final result to disk.

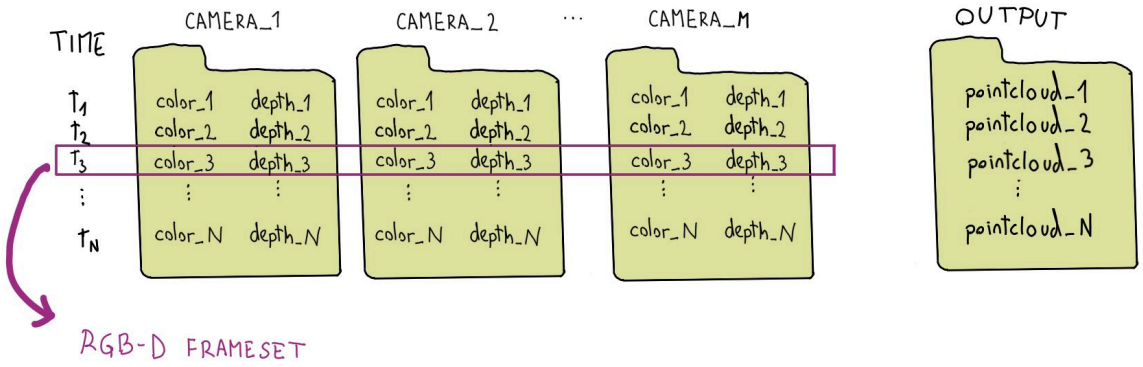

Figure 8.5: Inputs and outputs to the data processing software. RGB-D frames from every camera are stored in separate folders. Output of the processing is a sequence of three-dimensional pointclouds.

Every camera stores its sequence of RGB-D frames (color and depth frame pairs) in a separate directory, with timestamps or sequence numbers encoded in the filename. The processing software outputs a single three-dimensional frame[3] for every RGB-D frameset, as shown in image 8.5. An RGB-D frameset is a set of RGB-D frames, taken at the same time instant with exactly one RGB-D frame from each camera. Visualization of the output can be done using any software supporting the .ply format, such as MeshLab or Blender.

The data processing software is implemented in Python programming language, using Numpy and Open3D libraries for pointcloud processing operations. Open3D is a modern open-source library which implements functions for dealing with 3D data [49] and Numpy is an open-source library enabling efficient numerical computing with Python, widely used in the scientific community [15]. The selection of programming language and libraries allows processing of data on any computer platform. The outline of processing operations is presented in algorithm 2:

Algorithm 2 Processing pseudocode

```
1:  for frameset = 1, 2, ..., N do          // N = number of RGB-D framesets
2:      result ←null;
3:      for each rgbd_frame ∈ frameset do
4:          Parse a color and a depth frame pair (rgbd_frame)
5:          Register depth to color
6:          Project resulting RGB-D frame to a 3D pointcloud
7:          Transform pointcloud
8:          Append it to result
9:      end for
10:     Write result to disk
11: end for
```

Every color and depth frame pair is first parsed and aligned (lines 4 and 5 in algorithm 2). The color and depth sensors of a RealSense device have different, partially overlapping, fields of view which needs to be compensated for by registering depth frame to the color frame. This is done using depth-to-color extrinsic matrix obtained during camera calibration (chapter 6.3).

Next, a single RGB-D frame (2D) is projected into a textured point cloud (3D), using the intrinsic camera parameters obtained in calibration phase (6). The intrinsics of the RGB sensor are used for

[3]A single file containing aligned pointclouds from all cameras, stored in a .ply file format

pointcloud projection, since the depth frame was aligned to the color frame.

During the pointcloud formation, points with a depth value larger than 0.5m are being discarded in order to prevent cluttering the resulting pointcloud with background and only focus on the foot.

Finally pointcloud is transformed to a common coordinate frame, using the camera transformation matrix obtained in calibration phase from chapter 7.3. Pointclouds from all cameras are accumulated in a single variable, *result*, which is saved to disk with its sequence number encoded in the filename (as shown in image 8.5).

8.4. Limitations of centralized system architecture

Centralized approach of connecting multiple cameras to a single computer offers an intuitive and convenient solution to the problem. Most of the modern computers already contain all the required hardware for multi-camera acquisition, assuming a high-quality USB interface and an SSD disk.

Using a single computer for managing multiple cameras avoids most of the networking and synchronization challenges of distributed solutions and efficiently utilizes the hardware for both data acquisition and processing. However, there are limitations in terms of scalability and flexibility of the system: increasing the number of cameras, linearly increases the bandwidth requirements which quickly reach the limits of commodity hardware. Besides numerous potential bottlenecks further described in chapter 10 (e.g. USB interface, CPU processing), a limited amount of PCI expansion slots on consumer motherboards prevents adding USB interfaces indefinitely. Furthermore, the usage of cables for connecting the cameras with the computing platform limits the flexibility of camera positioning.

An alternative system architecture which addresses these limitations is discussed in chapter 12.

9

Scanning Results

9.1. 4D foot scans

The 4D foot scanner prototype described in chapter 8 outputs a sequence of three dimensional pointclouds, shown in Figure 9.1. Each frame in the sequence (Figure 9.1) contains pointclouds of a foot, captured with 4 cameras, observed from an arbitrary point of view in 3D space.

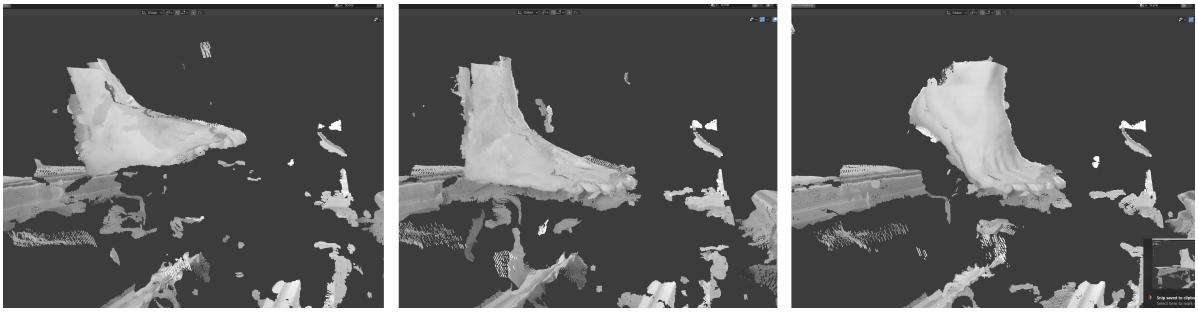

Figure 9.1: Three frames extracted from a scanning sequence, obtained with four cameras capturing the foot from the top side.

Besides removing points beyond a fixed threshold (0.5m), there is no additional pointcloud segmentation or filtering performed in the data processing stage (Chapter 8.3). This is why the resulting scan includes not only the foot, but parts of the background as well (parts of the walking stage and reflections from the plexiglass).

The overlapping regions between two camera's pointclouds are not perfectly aligned and stacked on top of eachother, which results in different measurements representing the same part of the foot. This issue was not resolved during our work, but can be avoided by improving on extrinsic camera calibration accuracy and applying fine-grained registration algorithms.

In order to observe contributions from individual cameras, a single frameset was extracted from the foot scanning sequence, displaying the pointclouds obtained from each camera separately, shown in Figure 9.2.

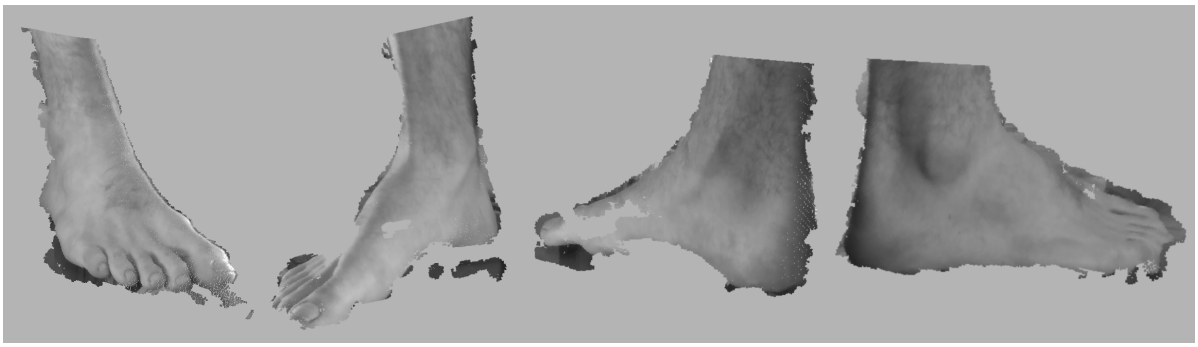

Figure 9.2: Single frame of a dynamic foot scan, where each camera's view is shown separately.

9.2. Static reconstruction accuracy

The reconstruction accuracy of the prototype was evaluated by scanning a plastic model of a hand and comparing the resulting scan with the original 3D model. Results obtained with our scanner are only coarsely aligned and do not involve pointcloud segmentation (extracting foreground points, the foot, from the background). This was compensated for, by manually cleaning the resulting scans and applying fine-grained pointcloud registration, using Geomagic Design X software for 3D data processing. Finally, the processed scan of the hand model was converted to a mesh and registered on top of the 3D model representing the ground truth. Figure 9.3 shows the analysis of deviation between the two 3D meshes, which spans errors ranging from -2.8 mm to +4.9 mm, with an average deviation of 0.58 mm.

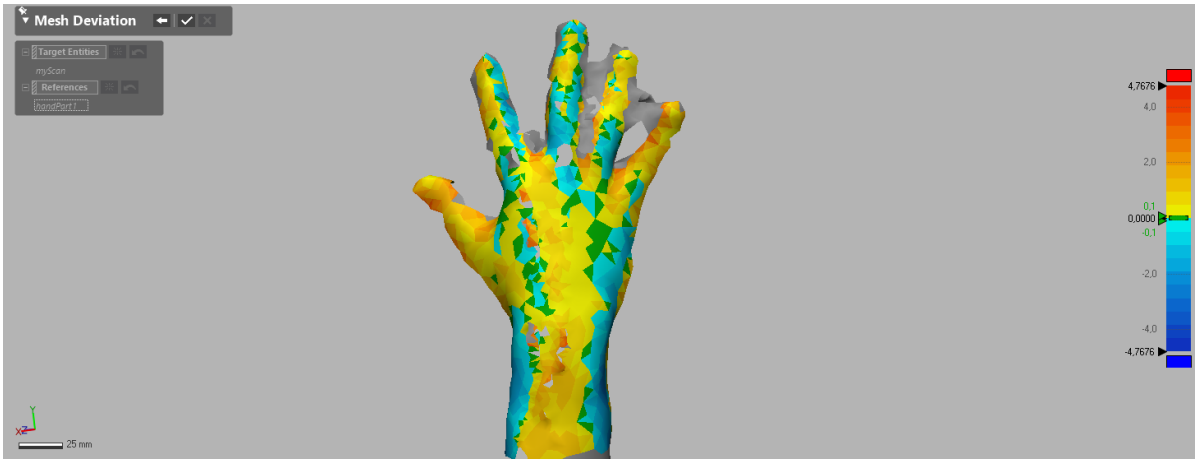

Figure 9.3: Accuracy evaluation of a static 3D scan obtained with our prototype. Colors represent the deviations between the scanning result and the ground truth, with average deviation of 0.58 mm. Data was obtained using Geomagic X software.

However, the resulting 3D reconstruction of the hand is not accurate due to poor alignment and noise in the pointclouds (e.g. gray surface between the fingers in Figure 9.3). This can be improved by performing additional processing of the scanned pointclouds.

In order to fully characterize the accuracy of our 4D foot scanning prototype, more measurements need to be performed in both static and dynamic scenarios, using real human feet.

10

Bandwidth measurements and bottlenecks

10.1. Overview

The RealSense D435i cameras are producing large amounts of data which might exceed the processing capabilities of typical commodity hardware. Increasing the framerate above 15 fps results in significant number of dropped frames and occasional camera failures, when streaming RGB-D data from 4 RealSense cameras on a single computer.

Several experiments were performed in order to investigate this limitation and quantify the hardware and software limitations of a single computer in multi-camera scenarios. In sections 10.2 and 10.3 we focus on USB data transport, measuring camera's bandwidth consumption and USB interface bandwidth capacity. Discussion of data acquisition bandwidth limitations is presented in Chapter 10.4.

10.2. USB theoretical background

USB is a serial bus connecting multiple peripheral devices to a host computer, using a tiered star topology [40] illustrated in Figure 10.1.

Communication on the bus is controlled (polled) by a single device, called the *host controller (HC)*, which has an integrated *root hub* with one or multiple *ports*. The bandwidth is shared between the ports, reaching theoretical data rate of 5 Gbit/s per bus in case of USB 3.1 Gen 1 (used by RealSense cameras). A single computing platform can contain multiple USB buses, with an examples shown in Figure 10.2.

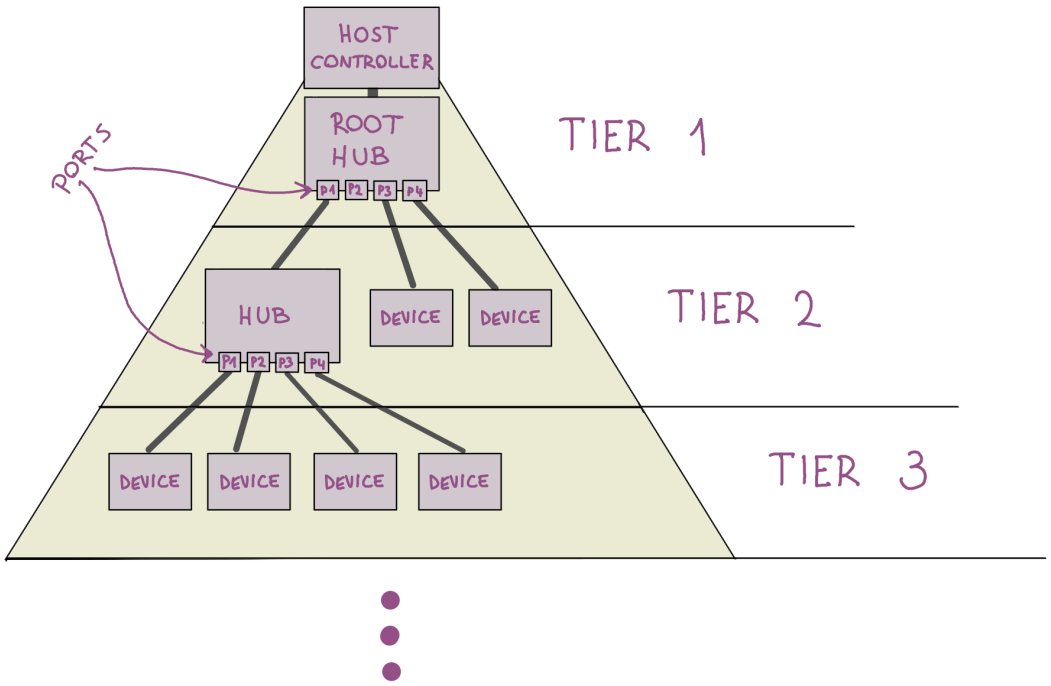

Figure 10.1: Physical topology of the USB bus

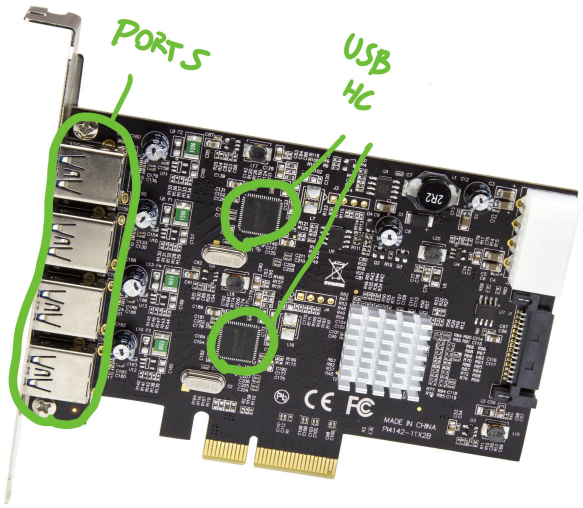

Figure 10.2: Example of an USB 3 PCIe expansion card, with a twp buses (two USB Host controllers) and four ports.

10.3. Measuring USB data troughput

The prototype described in Chapter 8, contains an USB 3.1 Gen 1 PCIe expansion card with four ports driven by two host controllers (two ports per bus).

The USB bandwidth consumption of a single RealSense camera depends on the resolution, framerate and pixel format of its video streams. The color sensor uses YUY2, a compressed pixel format using YUV color encoding system and chroma subsampling for reduced bandwidth consumption (compared to raw RGB formats). Depth sensor uses Z16 format for representing raw depth data. Both formats use 16-bits per pixel.

Streaming RGB-D data at highest settings[1] results in approx. 782 Mbit/s per camera, calculated according to the equation 10.1 below (where N indicates the number of enabled video streams):

$$device\_data\_rate[bit/s] = \sum_{i=1}^{N} n\_pixels_i * bits\_per\_pixel_i * framerate_i \tag{10.1}$$

Calculating the data rate using the formula above (10.1) only takes into account the video payload, while there might be additional communication overhead involved (e.g. metadata, control). In order to verify our calculations, the actual bandwidth consumption of a single RealSense camera was measured using Wireshark and *usbmon*. The *usbmon* kernel module allows passive monitoring of the USB bus with minimal processing overhead [46] and can be used with Wireshark, a popular open-source communication protocol analysis and monitoring tool.

Framerate [fps]	**Calculated bandwidth consumption [Mbit/s]**	**Measured bandwidth consumption [Mbit/s]**
6	78	79
15	195	196
30	391	392
60	782	778

Table 10.1: Theoretical and actual bandwidth consumption of a single RealSense D435i camera, streaming color and depth data at 848 x 480 px at different framerates.

From table 10.1 we can observe that bandwidth measurements closely match the calculated values and a constant overhead of 1 Mbps for all framerates (less than 1% overhead in 30 fps case).

In order to estimate the maximum troughput of a single USB 3.1 Gen 1 Host Controller, we connected four cameras on a single USB bus and enabled an RGB-D stream every 10 seconds, gradually increasing the bandwidth utilization. Data in Figure 10.3 was obtained by adapting the Algorithm 1, which now increases the counter for every received RGB-D frame and performs no additional processing.

From the measurements in Figure 10.3 we can observe that the maximal troughput does not reach the expected 3128 Mbit/s, but stops increasing at 1955 Mbit/s. Bandwidth consumption increased linearly for the first two cameras, while addition of third and fourth cameras resulted in only slight bandwidth increase and large number of dropped frames. Experiment was repeated using official software (RealSense Viewer) and showed similar results.

Since the actual USB data troughput stayed well below its theoretical maximum (39 %), it is unlikely that the USB bus saturation is the problem behind dropped frames. This hypothesis was confirmed by reading data[2] from an external SSD drive connected to the same USB host controller, which reached a higher troughput of approximately 3500 Mbit/s.

[1] 848 x 480 px resolution and 60 fps framerate are the highest common settings for both color and depth sensors.

[2] Using a benchmarking program *hdparm*, reading data directly from the disk (with operating system data caching disabled) and measuring its troughput.

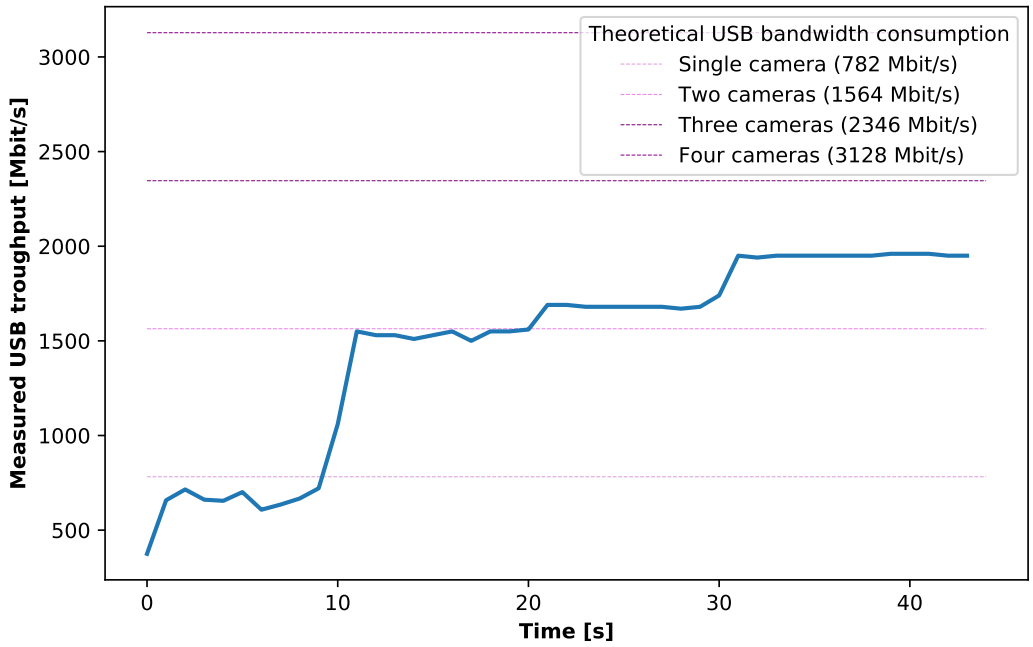

Figure 10.3: Gradually increasing USB bandwidth consumption, by enabling a camera every 10 seconds. First two cameras increase the bandwidth linearly, while enabling third and fourth cameras results in mostly dropped frames.

10.4. Discussion

From the experiments in previous chapter (10.3), we can observe that the USB bandwidth is not the main limitation when using multiple RealSense cameras on a single computer. However, further performance analysis is required in order to pinpoint the issue between the kernel drivers and the implementation of RealSense API.

These limitations could be addressed by circumventing the RealSense API and using the cameras with native operating system APIs or writing custom camera drivers from scratch. The RealSense cameras comply to the UVC 1.5 (Usb Video Class device) specification [41], which defines and standardizes video streaming functionality of USB devices. On Linux operating systems, there are generic drivers for UVC video devices available within the Video4Linux framework (V4L).

11

Camera synchronization

11.1. Overview

Synchronous data acquisition is important for accurate 3D reconstruction of dynamic scenes, when multiple image sensors are involved. In case of an RGB-D camera, the delay between the color and depth sensor (Chapter 11.2) results in an offset between the 3D data and its texture, while acquisition delays between multiple devices (Chapter 11.3) prevent accurate alignment of pointclouds.

RealSense API provides access to frame metadata, which contains multiple timestamps recorded during the acquisition process, shown in Figure 11.1 and summarized in Table 11.1. Synchronization delays between cameras are measured using timestamps recorded at the time[1] of image sensor readout on the camera (event A on Figure 11.1 and Table 11.1).

In order to provide a common time reference, the camera's clock is being synchronized with host computer's system clock, by constantly measuring message round-trip delays and estimating its time offset (similarly to the NTP synchronization algorithm). This feature is implemented in RealSense API, which provides all timestamps expressed in host computer's clock.

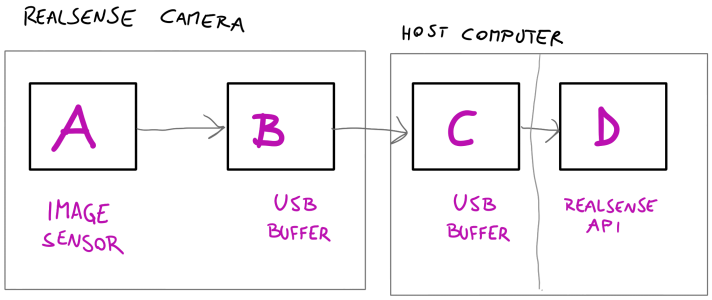

Figure 11.1: Diagram illustrating the lifetime of a frame and events (timestamps) which are recorded in the metadata. Event A is recorded during image sensor readout. Event B designates beginning of the USB data transfer towards the host computer. Event C is recorded when frame is read into the kernel buffers of the USB driver. Event D marks the arrival of frame to the userspace, waiting to be read by application.

Event	Timestamp name	Description
A	SENSOR_TIMESTAMP	image sensor readout (taken at the middle of exposure time)
B	FRAME_TIMESTAMP	beginning of the frame transmission
C	BACKEND_TIMESTAMP	time of frame arrival to the kernel buffers on the host computer
D	TIME_OF_ARRIVAL	time of frame arrival to the application (kernel to userspace transition)

Table 11.1: Summary of timestamps available in RealSense RGB-D metadata and their relations to events, marked in Figure 11.1

[1] Time at the middle of sensor exposure.

11.2. Color and depth frame synchronization

Ideally, an RGB-D camera captures data from all image sensors simultaneously while minor snychronization offsets might be introduced due to different exposure times. However, the RGB sensor of the RealSense D435i camera is mounted on a separate PCB and does not share the clock with depth (IR) image sensors[2], which prevents accurate synchronization between sensors.

In order to quantify the synchronization delays of color and depth sensors on a single camera, we observed timestamp differences between color and depth frame pairs. Matching of color and depth frames according to smallest difference in their timestamps is implemented in RealSense API. Results of a single camera streaming RGB-D data at 848 x 480 px at different framerates are shown in Table 11.2 (mean and std. deviation is calculated over 10 frames).

Framerate (fps)	Mean (ms)	Standard deviation (ms)
6	0.370	0.003
15	0.044	0.001
30	0.017	0.001
60	N.A.	N.A.

Table 11.2: Color vs depth acquisition delays

[2] https://community.intel.com/t5/Items-with-no-label/Hardware-Sync-of-Color-and-Depth-in-D435/m-p/530425/highlight/true#M7451

11.3. Inter-camera synchronization delays

Synchronization delays between individual cameras are larger and more significant in our application, compared to inter-sensor delays of an individual device. The delay between cameras depends on acquisition hardware and software, while the delays within the camera are imposed by their hardware design can not be influenced by users.

The best possible synchronization between cameras can be achieved by providing a reference clock to their dedicated hardware pins (e.g. using an external signal generator or using one of cameras' clocks as a reference). This method of camera synchronization, called genlocking, is commonly used in video broadcasting industry and possible to use on RealSense cameras [22] [21], using the connector shown in image 11.2. However, this method was not used in our work due to limited software support and known hardware limitations on D435i model [3].

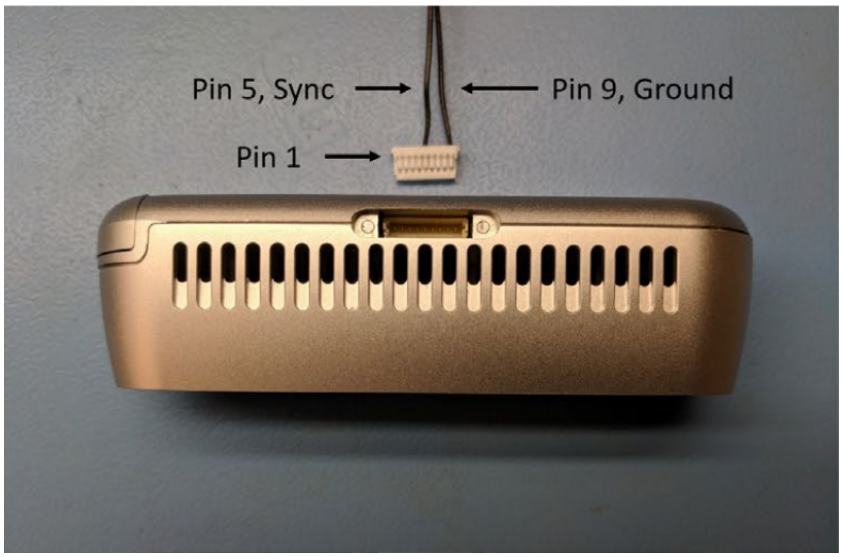

Figure 11.2: Connector used for hardware synchronization of clocks between multiple RealSense devices. (Image source: [22])

Synchronous start of RGB-D streaming on all cameras, described in chapter 8.2, provided sufficient synchronization for low framerates (up to 15 fps). This was measured by comparing depth sensor timestamps, using two RealSense cameras connected to a single computer, streaming RGB-D data at 15 fps.

For every pair of corresponding depth frames, the absolute difference ($diff$) between their timestamps (t_A and t_B) was calculated: $diff = |t_A - t_B|$. Table 11.3 shows the mean and standard deviation of each experiment, which was repeated 10 times.

[3] Clock signals provided trough the connector on Figure 11.2 are not connected to the RGB sensor. This hardware limitation allows synchronizing depth sensors only and is not present in other RealSense camera models (D415 and D455).

Table 11.3: Measurements of synchronization accuracy between two RealSense depth sensors, streaming 10 seconds of data at 15 fps and calculating mean of absolute timestamp differences and their standard deviation in milliseconds. Experiment was repeated 10 times.

Measurement	Mean [ms]	Standard deviation [ms]
1	22.5	1.6
2	19.6	15.1
3	13.9	6.4
4	22.5	6.3
5	9.8	6.9
6	7.0	4.3
7	5.5	4.0
8	12.3	7.8
9	16.5	7.1
10	13.6	11.7

12

Modular system architecture: proof of concept

12.1. Overview

In order to overcome the limitations imposed by the centralized architecture of our prototype (chapter 8), an alternative, modular solution was investigated. Instead of connecting all cameras to a single computer, there is a now a dedicated single-board computer providing hardware resources for data acquisition and storage. After data is captured by each module independently, frames from all cameras are gathered, post-processed and visualized on a central computer. This can be an arbitrary computer with sufficient storage capacity, using an arbitrary network connection with the acquisition modules, since there are no timing constraints in the offline processing system. Comparison between centralized and modular system architecture is shown in Figure 12.1.

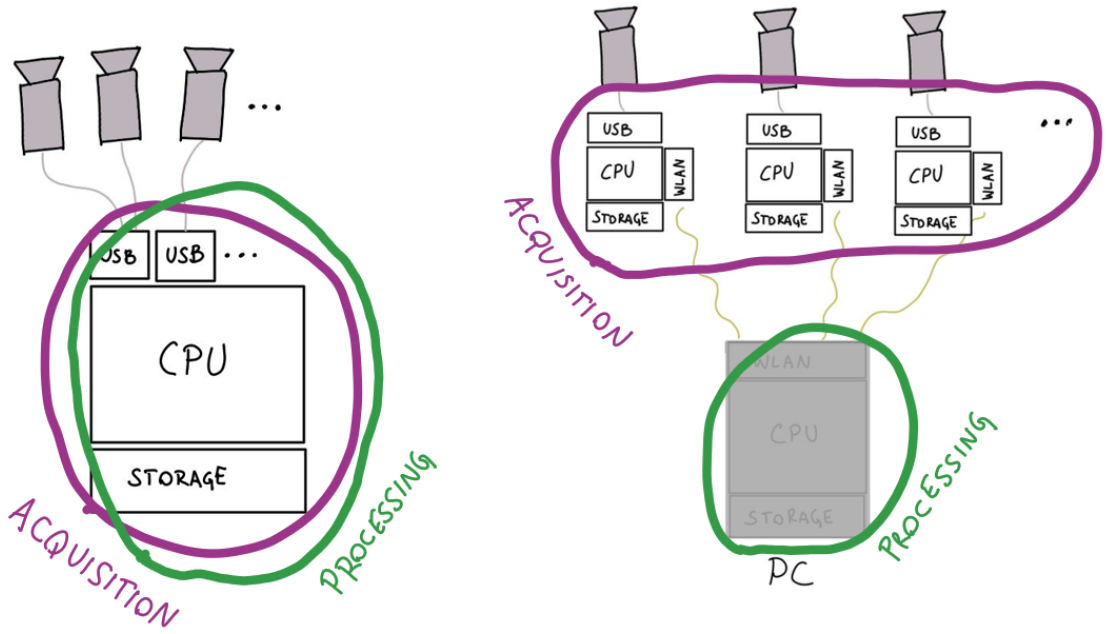

Figure 12.1: Comparison of centralized and modular system architectures.

This architecture eliminates the problem of scalability by separating data acquisition and data processing not only in software, but in hardware as well. The number of devices in centralized scenario was limited due to hardware limitations related to data acquisition, which is not the case with acquisition modules. Furthermore, flexibility limitations caused by USB cables can be solved if the acquisition module is battery-powered and wirelessly connected to the central computer.

12.2. Acquisition module: RealSense D435i + Raspberry Pi 4 + SSD

Our implementation of this system is based on acquisition modules, which consist of a single RealSense D435i camera, Raspberry Pi 4 single board computer and an SSD drive, shown in Figure 12.2. Modules are running the data acquisition software described in section 8.2 on a linux operating system (Raspberry Pi, previously called Raspbian).

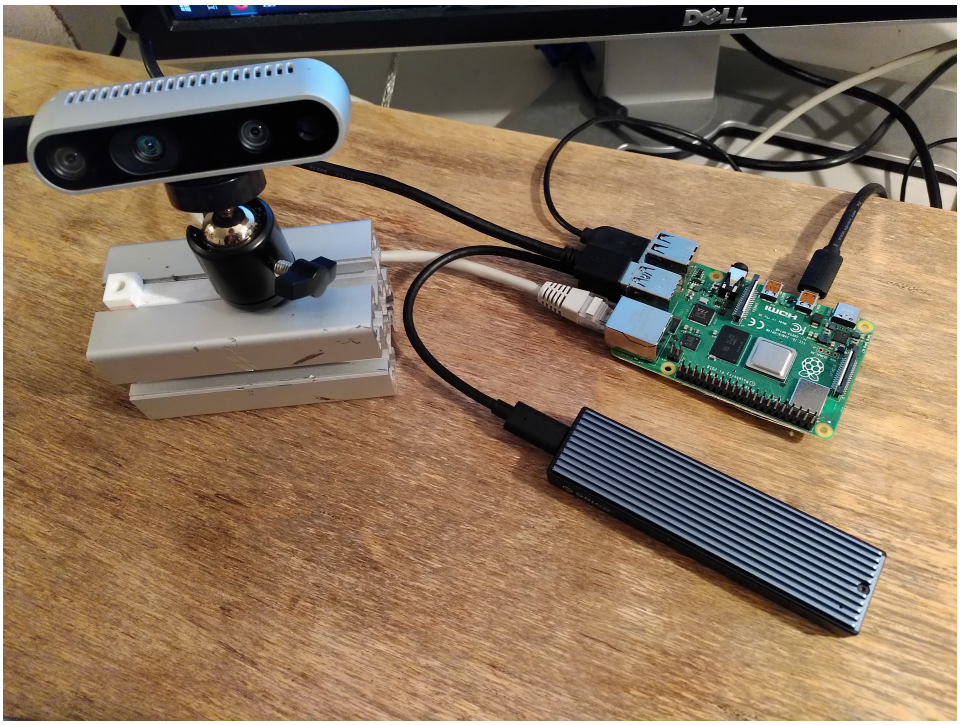

Figure 12.2: Image of a single acquisition module of the modular prototype. A RealSense camera is connected to a Raspberry Pi 4 single-board-computer, powered by an external SSD drive.

Raspberry Pi computers are normally booted from an SD card, which have smaller storage capacity and transfer speed in comparison to SSD or HDD drives used in desktop computers. While storage capacity is not the primary concern in our case, due to short recording sequences (less than a minute), transfer speeds of SD cards would pose a bottleneck in acquisition framerate. A single RealSense camera produces approx. 98 MB/s of data which would exceed the writing speeds of an SD card. A performance comparison using Raspberry Pi 4 showed that using an SSD drive connected via USB 3 can reach approx. 222 MB/s sequential write speeds [18], while SD cards can not reach write speeds higher than 40 MB/s [17]. In order to achieve higher framerates, an SSD drive[1] was used for both frame capturing and the operating system.

An USB enclosure is required in order to connect an SSD drive to the USB 3 ports of the Raspberry Pi 4, however not all enclosures offer the same performance. Data troughput greatly depends on the PCIe to USB bridge chipset used within the enclosure, specially its UASP protocol implementation [19] which is not implemented in all chipsets and contains bugs in some (widespread) chipsets [27]. Furthermore, the bootloader firmware of the Raspberry Pi casuses a short power-drop on the USB ports, which resets the device and disrupts the booting process occasionally. This issue is not present in all USB enclosures and can be resolved by using a powered USB hub [2].

[1] Kingston A2000, an M.2 interface SSD drive with 250 GB storage capacity

[2] The USB enclosure used in our experiments uses the RTL9210 chipset, which caused no problems during the boot process.

In order to boot a Raspberry Pi 4 from an USB storage device, the bootloader version needs to supports USB boot and correct boot order needs to be configured. At the time of performing our experiments, this feature was not yet officially supported and required manually flashing the on-board EEPROM memory with the latest bootloader version and changing its configuration. However, as the feature is now officially supported [16], the procedure is relatively straightforward and essentially ”plug-and-play” .

12.3. Performace limitations

Single board computers are suitable for implementing acquisition modules due to their compact form and low price, but might introduce several limitations in terms of data acquisition performance. Running the RGB-D acquisition software, described in chapter 8.2, allowed capturing frames at 15 fps which is only a sixth of the framerate achieved on a desktop computer. However, the main reason for this is not hardware, but inefficient software backend used in Raspberri Pi operating systems.

RealSense API depends on certain kernel functionality[3] which is present on desktop computers (e.g. Ubuntu), but not on single board computers (e.g. Raspberry Pi OS). In order to support a wider range of computer platforms, the RealSense developers provide an alternative software backend which implements all necessary functionality in user space, avoiding the need for non-trivial modifications[4] of the operating system. Figure 12.3 shows comparison between the default backend using kernel drivers (called V4L_BACKEND) and the alternative backend (called RSUSB_BACKEND).

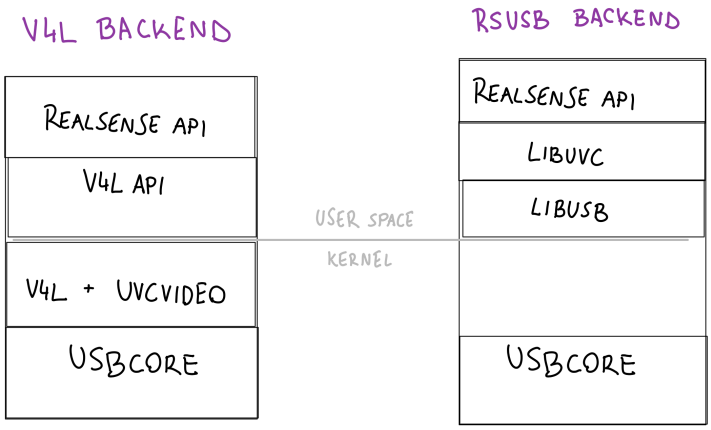

Figure 12.3: Comparison of software components used in different RealSense backends.

While providing a simpler installation method, the user space implementation of video streaming functionality comes with a performance cost, due to increase in processor context switches caused by system calls. In order to compare the performance of two backends, we captured 10 seconds of RGB-D data on a desktop computer and measured the time spent executed system calls using the *strace* utility, shown in table 12.1. It can be observed that the user space backend implementation (RSUSB_BACKEND) spent nearly double amount of time executing system calls which decreases the performance and limits the maximum framerate.

In order to evaluate the limits of the hardware used in acquisition modules, software limitations need to be addressed first. Experiments using the native software backend (V4L_BACKEND) of the RealSense API or custom drivers could provide further insight, but were not included in the scope of our work.

Experiments with the JMS583 based enclosure caused occasional operating system freezes (UASP imlpementation bugs) and boot process failures (power delivery issues).

[3]The V4L (Video 4 Linux) and uvcvideo kernel modules. The uvcvideo module needs to be patched, as it does not support the depth video formats and metadata by default.

[4]Enabling RealSense API on Raspberry Pi OS requires modifying and compiling the linux kernel, which is a complex process and is not officially supported by RealSense.

RealSense backend	number of calls	total time (s)
V4L (kernel)	19473	5.979502
RSUSB (user space)	64663	11.274009

Table 12.1: Comparing usage of system calls between the two RealSense backend implementations. User space implementation of the backend (RSUSB) spends almost double amount of time executing system calls compared to the kernel backend implementation (V4L).

12.4. Synchronization between modules

Synchronized data acquisition in the modular scenario is more challenging compared to the centralized prototype, described in Chapter 8. There are now multiple independent computers which need to ensure synchronous operation of RealSense cameras in order to obtain accurate 4D reconstruction of dynamic scenes.

Simultaneous start of RGB-D streaming on all modules is achieved similarly to previous prototype. The *streaming* flag (in Alogrithm 1) is now controlled, by a hardware interrupt generated by a pushbutton which is connected to GPIO pins of all modules. This can be replaced with a wireless solution in future, allowing greater flexibility when positioning the modules.

Further measurements are required in order to evaluate if this kind of synchronization between modules is sufficient for data acquisition at framerates of 15 fps or higher.

13

Conclusions

13.1. Summary

In this graduation work, we developed a 4D foot scanning prototype using four RGB-D cameras connected to a single computer, allowing RGB-D data acquisition at 15 fps with resolution of 848x480 pixels.

The first part involved a literature study of existing 3D and 4D foot scanners, followed by an in-depth examination of the RealSense D435i camera accuracy and optimal parameter configuration. Existing camera calibration techniques were researched and a simple extrinsic calibration method was used for solving the pointcloud alignment problem.

Upon solving the challenges of camera calibration and pointcloud alignment, the 4D scanner prototype was built and used for testing our implementation of the 4D reconstruction pipeline. A walking stage with a plexiglass section was built, allowing the capture of a single step from all sides, including its bottom. Cameras were mounted on a separate frame with custom 3D printed mounts. The resulting prototype allows a 4D reconstruction of a foot in motion.

Second part of our work focused on embedded system design and data acquisition challenges, addressing the limitations of centralized system architecture. Upon an investigation of data acquisition bottlenecks and camera synchronization measurements, an alternative system architecture was proposed. Modular 4D scanning prototype concept was (partially) implemented using multiple single board computers, allowing data acquisition at 15 fps and 848 x 480 px resolution.

13.2. Future work and potential improvements

The interdisciplinary nature of this work allows plenty of room for improvement, since an in-depth study of all its aspects would be impossible.

While our accuracy evaluation confirmed the suitability of the Intel RealSense D435i camera for 4D foot scanning purposes, further experiments need to be performed in order to obtain a more representative characterization of the device. Replicating the measurements performed by Carfagni et al. [11] would allow fair comparison of the D435i with other RGB-D devices (such as D455, Kinect and others).

There are multiple weaknesess in our methods of camera calibration and pointcloud alignment, which can be addressed and might significantly improve the scanning results. Intrinsic parameters of RGB-D cameras were calibrated using the default manufacturer's software tools, which often produce sub-optimal calibration results and are not scientifically evaluated. The pairwise calibration of cameras is prone to error accumulation since there is only one line of camera relations used, which could be improved by taking into account all possible camera pair combinations and distributing the accumulated error. An alternative method of extrinsic camera calibration (e.g. sphere object calibration [34]) would allow a more convenient prototype usage in practice. Furthermore, there was no fine pointcloud registration applied in our post-processing software (such as ICP algorithm) which would improve scanning results.

According to our investigation, the RealSense API posed the biggest limitation in terms of data acquisition. It is a cross-platform library providing convenient access to RealSense cameras and expecting high-performance and efficiency would be unfair, specially in resource-limited, embedded com-

puters. Since the cameras comply with the UVC 1.5 standard specification, these limitations can be addressed by implementing custom camera drivers which can efficiently utilize all available hardware resources on a specific computing platform.

Finally, the scanning prototype needs to be evaluated in terms of foot reconstruction accuracy in both static and dynamic scenarios, using multiple subjects.

Bibliography

[1]

[2] Foot deformation during walking: Differences between static and dynamic 3D foot morphology in developing feet, 2014. URL http://dx.doi.org/10.1080/00140139.2014.899629.

[3] Sweeping-based volumetric calibration and registration of multiple RGBD-sensors for 3D capturing systems. *Proceedings - IEEE Virtual Reality*, pages 167–176, 2017. doi: 10.1109/VR.2017.7892244.

[4] LCC 3dMD. 3dmdfoot™ system series. URL https://3dmd.com/products/#!/foot.

[5] Alfredo BALLESTER, Ana PIEROLA, Eduardo PARRILLA, Mateo IZQUIERDO, Jordi URIEL, Beatriz NACHER, Vicent ORTIZ, Juan C. GONZALEZ, Alvaro PAGE, and Sandra ALEMANY. Fast, Portable and Low-Cost 3D Foot Digitizers: Validity and Reliability of Measurements. (October):218–225, 2017. doi: 10.15221/17.218.

[6] Bettina Barisch-Fritz, Timo Schmeltzpfenning, Clemens Plank, and Stefan Grau. Foot deformation during walking: Differences between static and dynamic 3D foot morphology in developing feet, 2014. ISSN 13665847. URL http://dx.doi.org/10.1080/00140139.2014.899629.

[7] Stephan Beck and Bernd Froehlich. Volumetric calibration and registration of multiple RGBD-sensors into a joint coordinate system. *2015 IEEE Symposium on 3D User Interfaces, 3DUI 2015 - Proceedings*, pages 89–96, 2015. doi: 10.1109/3DUI.2015.7131731.

[8] Abhishektha Boppana and Allison Anderson. DynaMo: Dynamic Body Shape and Motion Capture with Intel RealSense Cameras. *Journal of Open Source Software*, 4(41):1466, 2019. ISSN 2475-9066. doi: 10.21105/joss.01466.

[9] Abhishektha Boppana and Allison P. Anderson. Dynamic foot morphology explained through 4D scanning and shape modeling. 2020. URL http://arxiv.org/abs/2007.11077.

[10] Sicco Bus, G Valk, Robert van deursen, David Armstrong, Carlo Caravaggi, Petr Hlavacek, Karel Bakker, and P Cavanagh. The effectiveness of footwear and offloading interventions to prevent and heal foot ulcers and reduce plantar pressure in diabetes: A systematic review. *Diabetes/metabolism research and reviews*, 24 Suppl 1:S162–80, 05 2008. doi: 10.1002/dmrr.850.

[11] Monica Carfagni, Rocco Furferi, Lapo Governi, Chiara Santarelli, Michaela Servi, Francesca Uccheddu, and Yary Volpe. Metrological and critical characterization of the intel D415 stereo depth camera. *Sensors (Switzerland)*, 19(3), 2019. ISSN 14248220. doi: 10.3390/s19030489.

[12] Intel Corporation. Intel® realsense™ d400 series calibration tools. URL https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/RealSense_D400_Dyn_Calib_User_Guide.pdf.

[13] Intel Corporation. Which device is right for you, 2019. URL https://www.intelrealsense.com/which-device-is-right-for-you/.

[14] Térence Coudert and Pierre Vacher. A method to obtain 3D foot shape deformation during the gait cycle. *,9th International Symposium on the 3D analysis of Human*, pages 3–6, 2006. URL http://www.univ-valenciennes.fr/congres/3D2006/Abstracts/117-Coudert.pdf.

[15] Numpy Steering Council. About us (numpy). URL https://numpy.org/about/.

[16] Raspberry Pi Foundation. Raspberry pi boot modes, 2020. URL https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/msd.md.

[17] Jeff Geerling. Raspberry pi microsd card performance comparison - 2019, 2019. URL https://www.jeffgeerling.com/blog/2019/raspberry-pi-microsd-card-performance-comparison-2019.

[18] Jeff Geerling. The fastest usb storage options for raspberry pi, 2020. URL https://www.jeffgeerling.com/blog/2020/fastest-usb-storage-options-raspberry-pi.

[19] Jeff Geerling. Uasp makes raspberry pi 4 disk io 50% faster, 2020. URL https://www.jeffgeerling.com/blog/2020/uasp-makes-raspberry-pi-4-disk-io-50-faster.

[20] Vicente Giménez, Marcelo Saval-Calvo, Jorge Azorin-Lopez, José Rodríguez, Miguel Cazorla, Sergio Orts, and Andrés Guilló. A comparative study of registration methods for rgb-d video of static scenes. *Sensors*, 14:8547–8576, 05 2014. doi: 10.3390/s140508547.

[21] Anders Grunnet-jepsen, Aki Takagi, John Sweetser, Tri Khuong, and Dave Tong. External Synchronization of Intel ® RealSense ™ Depth cameras.

[22] Anders Grunnet-Jepsen, Paul Winer, Aki Takagi, John Sweetser, Kevin Zhao, Tri Khuong, Dan Nie, and John Woodfill. Using the Intel ® RealSense TM Depth cameras D4xx in Multi-Camera Configurations. 2018. URL https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/RealSense{_}Multiple{_}Camera{_}WhitePaper.pdf{%}0Ahttps://www.digikey.com/product-detail/en/jst-sales-america-inc/SHR-09V-S/455-1399-ND/759888.

[23] Daniel Herrera, Juho Kannala, and Janne Heikkiï. Joint depth and color camera calibration with distortion. pages 144–147, 2016.

[24] Intel. D400 Series Product Family. (November):1–122, 2018. URL https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf.

[25] Makoto Kimura, Masaaki Mochimaru, and Takeo Kanade. Measurement of 3D foot shape deformation in motion. *PROCAMS 2008 - ACM/IEEE 5th International Workshop on Projector Camera Systems*, 2008. doi: 10.1145/1394622.1394636.

[26] Makoto Kimura, Masaaki Mochimaru, and Takeo Kanade. 3D measurement of feature cross-sections of foot while walking. *Machine Vision and Applications*, 22(2):377–388, 2011. ISSN 09328092. doi: 10.1007/s00138-009-0238-3.

[27] Nathan Kirsch. Jmicron jms583 – controller version matters for portable usb drives, 2020. URL https://www.legitreviews.com/jmicron-jms583-controller-version-matters-for-portable-usb-drives_219422.

[28] Reinhard Klette. *Concise Computer Vision - An Introduction into Theory and | Reinhard Klette | Springer*. 2014. ISBN 9781447163190. URL http://www.springer.com/us/book/9781447163190.

[29] Andrew Maimone, Jonathan Bidwell, Kun Peng, and Henry Fuchs. Enhanced personal autostereoscopic telepresence system using commodity depth camera. *Computers Graphics*, 36:791–807, 11 2012. doi: 10.1016/j.cag.2012.04.011.

[30] Boštjan Novak, Janez Možina, and Matija Jezeršek. 3D laser measurements of bare and shod feet during walking. *Gait and Posture*, 40(1):87–93, 2014. ISSN 18792219. doi: 10.1016/j.gaitpost.2014.02.015.

[31] Timo Schmeltzpfenning, Clemens Plank, Inga Krauss, Petra Aswendt, and Stefan Grau. Dynamic foot scanning: A new approach for measurement of the human foot shape while walking. *Footwear Science*, 1(June):28–30, 2009. ISSN 19424299. doi: 10.1080/19424280902977111.

[32] Kristina Stanković, Brian G. Booth, Femke Danckaers, Fien Burg, Philippe Vermaelen, Saartje Duerinck, Jan Sijbers, and Toon Huysmans. Three-dimensional quantitative analysis of healthy foot shape: A proof of concept study. *Journal of Foot and Ankle Research*, 11(1):1–13, 2018. ISSN 17571146. doi: 10.1186/s13047-018-0251-8.

[33] Kristina Stanković, Toon Huysmans, Femke Danckaers, Jan Sijbers, and Brian G. Booth. Subject-specific identification of three dimensional foot shape deviations using statistical shape analysis. *Expert Systems with Applications*, 151, 2020. ISSN 09574174. doi: 10.1016/j.eswa.2020.113372.

[34] Aaron N. Staranowicz, Garrett R. Brown, Fabio Morbidi, and Gian Luca Mariottini. Practical and accurate calibration of RGB-D cameras using spheres. *Computer Vision and Image Understanding*, 137:102–114, 2015. ISSN 1090235X. doi: 10.1016/j.cviu.2015.03.013. URL http://dx.doi.org/10.1016/j.cviu.2015.03.013.

[35] G. K. L. Tam, Z. Cheng, Y. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X. Sun, and P. L. Rosin. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1199–1217, 2013. doi: 10.1109/TVCG.2012.310.

[36] Scott Telfer and James Woodburn. The use of 3D surface scanning for the measurement and assessment of the human foot. *Journal of Foot and Ankle Research*, 3(1):1–9, 2010. ISSN 17571146. doi: 10.1186/1757-1146-3-19.

[37] Scott Telfer and James Woodburn. The use of 3D surface scanning for the measurement and assessment of the human foot. *Journal of Foot and Ankle Research*, 3(1):1–9, 2010. ISSN 17571146. doi: 10.1186/1757-1146-3-19.

[38] Ali K. Thabet, Emanuele Trucco, Joaquim Salvi, Weijie Wang, and Rami J. Abboud. Dynamic 3D shape of the plantar surface of the foot using coded structured light: A technical report. *Journal of Foot and Ankle Research*, 7, 2014. ISSN 17571146.

[39] Bonnie Yuk San Tsung, Ming Zhang, Yu Bo Fan, and David Alan Boone. Quantitative comparison of plantar foot shapes under different weight-bearing conditions. *Journal of Rehabilitation Research and Development*, 40(6):517–526, 2003. ISSN 07487711. doi: 10.1682/JRRD.2003.11.0517.

[40] Inc. USB Implementers Forum. Universal serial bus 3.0 specification, revision 1.0, 2011. URL https://web.archive.org/web/20160412211620/http://www.usb.org/developers/docs/usb_31_040816.zip.

[41] Inc. USB Implementers Forum. Universal serial bus device class definition for video devices, revision 1.5, 2012. URL https://www.usb.org/document-library/video-class-v15-document-set.

[42] Víctor Villena-Martínez, Andrés Fuster-Guilló, Jorge Azorín-López, Marcelo Saval-Calvo, Jeronimo Mora-Pascual, Jose Garcia-Rodriguez, and Alberto Garcia-Garcia. A quantitative comparison of calibration methods for RGB-D sensors using different technologies. *Sensors (Switzerland)*, 17 (2), 2017. ISSN 14248220. doi: 10.3390/s17020243.

[43] Frances K.W. Wan, Kit Lun Yick, and Winnie W.M. Yu. Validation of a 3D foot scanning system for evaluation of forefoot shape with elevated heels. *Measurement: Journal of the International Measurement Confederation*, 99:134–144, 2017. ISSN 02632241. doi: 10.1016/j.measurement.2016.12.005. URL http://dx.doi.org/10.1016/j.measurement.2016.12.005.

[44] Channa P. Witana, Shuping Xiong, Jianhui Zhao, and Ravindra S. Goonetilleke. Foot measurements from three-dimensional scans: A comparison and evaluation of different methods. *International Journal of Industrial Ergonomics*, 36(9):789–807, 2006. ISSN 01698141. doi: 10.1016/j.ergon.2006.06.004.

[45] Munan Yuan, Xiaofeng Li, Jinlin Xu, Chaochuan Jia, and Xiru Li. 3D foot scanning using multiple RealSense cameras. *Multimedia Tools and Applications*, (December 2018), 2020. ISSN 15737721. doi: 10.1007/s11042-020-09839-w.

[46] Pete Zaitcev. The usbmon: USB monitoring framework. *Linux Symposium*, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.3893&rep=rep1&type=pdf#page=299.

[47] Hechuan Zhang, Zhiyong Chen, Shihui Guo, Juncong Lin, Yating Shi, Xiangyang Liu, and Yong Ma. Sensock: 3D Foot Reconstruction with Flexible Sensors. *Conference on Human Factors in Computing Systems - Proceedings*, pages 1–13, 2020. doi: 10.1145/3313831.3376387.

[48] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. ISSN 01628828. doi: 10.1109/34.888718.

[49] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.