University of Twente

Master Thesis

Report: Targetless approach for Automated Calibration of Multi-Sensor Systems

Author:

Rob Binnenmars (S2322285)

Student, Master Robotics
University of Twente

Supervisor:

Dr. Ville Lehtola

Assistant Professor
Faculty of Geo-Information Science and Earth Observation

Advisor:

Muhammad Affan

PhD Candidate
Faculty of Geo-Information Science and Earth Observation

UNIVERSITY OF TWENTE.

December 16, 2024

Contents

Acknowledgements

Before starting the report, I want to thank my supervisor, Dr. Ville Lehtola. He supported me from all the way in February when deciding to go for this calibration topic, guiding me through the development and testing phases and giving me useful and needed feedback from the very beginning till the very end with writing the report. His weekly meetings helped me to keep on track with the project as much as possible. I'm truly grateful for all his time and effort.
Another person I want to thank for his support is, Affan, he acted more like an advisor to me. He helped me with setting up the initial code for the pointcloud generation, further refinement with the SLAM part of the Fast-LIO-SLAM, and later on with data capture with the handheld combination. Besides that, he also gave me useful information and papers to further the development of the project.
Besides Ville and Affan, I want to also thank my family for their support while doing the development and writing the report.

1 Introduction

1.1 Context

In recent years, there has been a significant increase in sensor systems that include both LiDARs and cameras. Examples are self-driving cars, drones, and robots, where these sensors are needed for understanding and interacting with their environment. A camera captures texture and color information by measuring light in the visible spectrum, while a LiDAR sensor operates in the (near) infrared spectrum to measure depth. The data from these sensors on their own are useful, but by combining the data more things can be retrieved from them. Cameras are used in detecting and classifying objects using color and texture, but determining the accurate size and distance of objects with one camera can be a difficult task. A LiDAR provides accurate depth information but struggles to classify objects where color and texture are necessary. By combining the data from both sensors, these systems can achieve better object detection, classification, and tracking, which then can be used for a whole lot of other algorithms or robots to determine the next task or movement.

In Figure 1, a pair of depth and camera images is shown. In the depth image, the lighter the color, the farther away the detected point is. Many objects and structures can be recognized from both images, for example, the trees and their branches and leaf coverage, the building and its balconies or walkways, and the car entrance of the building. However, shadows and finer details like color changes or road markings are not visible in the depth image, whereas they are easily seen in the camera image. The top of the building is also not visible in the depth image, but in the camera image, there is more detail. Hence, combining these two modalities makes it possible to better understand the scene, which is crucial for many applications.

As an example of a multi-sensor system, the Waymo Jaguar I-Pace RoboTaxi uses nine cameras, four LiDARs, and six radars to perceive its environment, detect, classify, and track objects like pedestrians and vehicles (Vision, 2021). These sensors are initially calibrated in a lab setting, but slight shifts over time require recalibration to maintain performance. Similarly, self-driving cars, drones, and other autonomous systems need accurate sensor calibration to perform tasks effectively. While the ideal solution is to mount the sensors on a rigid base to prevent any relative movement between them, this may not always be possible due to physical constraints. Additionally, sensors might become misaligned after collisions or impacts with other objects. For these reasons, an automatic recalibration approach that does not require targets is the most effective solution.

While the camera and LiDAR pairs could be recalibrated in a lab, this would mean that a person would need to manually move the system back to the lab, which takes time and money. But what if the sensor system is on a robot in an environment that is not reachable by humans, like the bottom of the ocean or on Mars for example? Then there needs to be a system to automatically recalibrate the sensors, since it is not possible to get it back to the lab easily so this automatic calibration also needs to be targetless.

In some applications, such as 3D environment mapping using sensor systems mounted on backpacks (for example Lehtola et al. (2017)), the sensors are used only to capture and record data, without being involved in any actions or interactions with the environment during the recording process. However, they still require accurate calibration to produce accurate digital copies of the environment. If LiDAR and camera data are not properly calibrated, misaligned depth and color information result in incorrect object localization, misclassified objects, or degraded SLAM and localization performance. Good extrinsic and intrinsic calibration is crucial for accurate data reconstruction, mapping, navigation, and object classification.

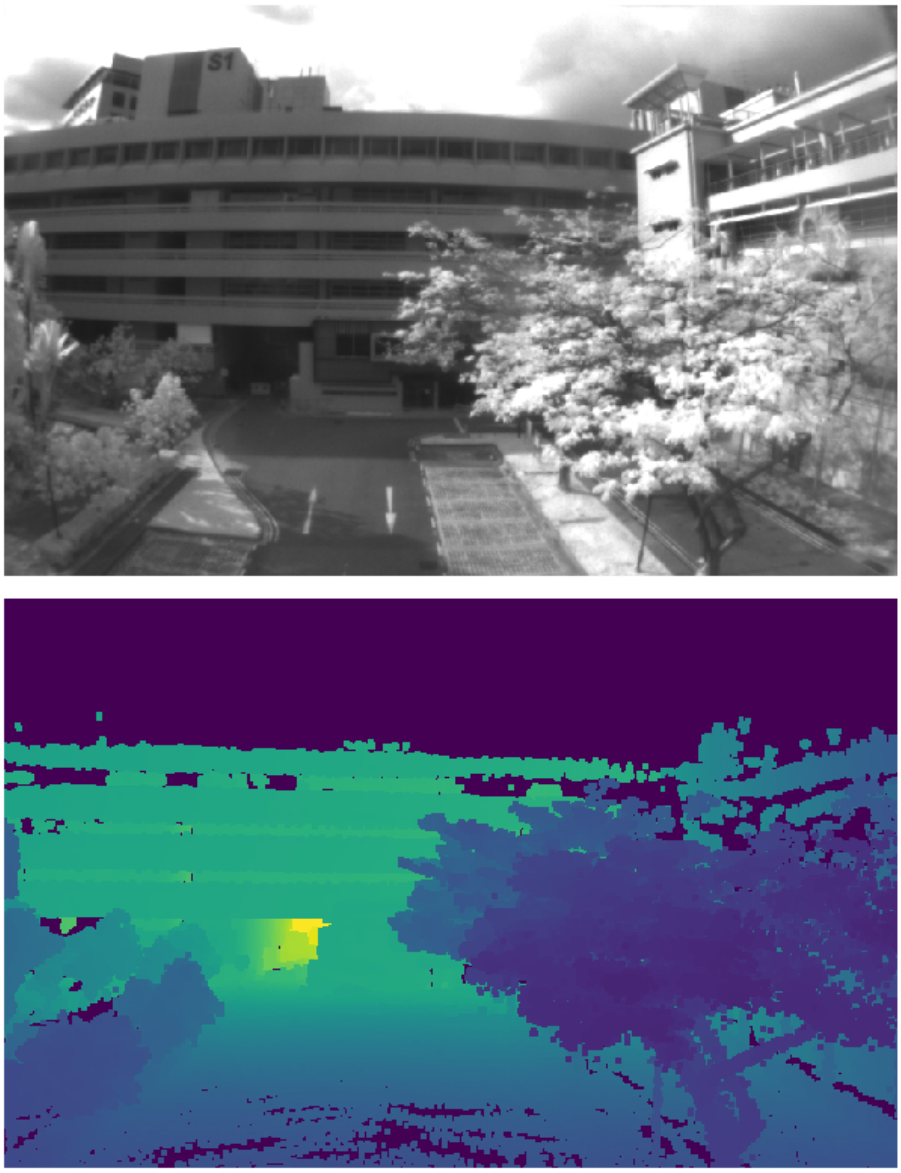

Figure 1: Depth image (bottom) and grayscale camera image (top) pair from the NTU eee1 dataset.

1.2 Problem Statement

Traditional techniques to determine the Extrinsic calibration between a LiDAR and a camera is to find corresponding points in both the camera image and the LiDAR frame. This can be done by manually selecting the points in both modalities that correspond, but this takes a lot of time and is not accurate. To improve that an automatic approach was developed to automatically get corresponding points, but to get accurate matching targets were used that can both be seen in the lidar data and the camera image. These targets are often reflective with a known dimension. This resulted in a more accurate calibration, but takes time to set the targets up. An improvement for this was to develop a targetless approach, these methods find features like straight edges that can be seen in both.
When the points are matched between both data modalities, an optimization algorithm is used to optimize the translation and rotation that gives the lowest reprojection error. So the best transformation that gives the lowest error between the matching points when reprojecting the points with the transform. More points will give a better transformation

but it can take more time to determine this transformation, depending on how accurate the feature matching is.
To fuse LiDAR and camera data, both intrinsic and extrinsic calibration are needed. While this process can be done manually, an automated approach for multiple sensors that can be done anywhere is an interesting research topic. The ideal automatic method would perform both intrinsic and extrinsic calibration without the need for targets, making the process faster.
In this thesis, we investigate the feasibility of an approach that seeks to achieve exactly that: automatic targetless calibration of both the camera's intrinsic parameters and the extrinsic transformation between the LiDAR and the camera. This is done by making an image from the lidar data and matching it to the camera image. The calibration process begins with the collection of Camera, LiDAR, and IMU data, with which the latter two are used to create a pointcloud through FAST-LIO-SLAM(Kim, 2024a), a combination of Fast-LIO (Zhu et al., 2022a) and SC-PGO(Kim, 2024b). This results in a dense pointcloud of the environment, IMU to LiDAR transformation, as well as the corresponding poses of the IMU within that pointcloud.
Next, the IMU poses are interpolated to get the exact IMU pose when each camera image is taken. A depth image is generated at the same pose as the camera, using an initial camera-to-IMU transformation, and intrinsic camera parameters. This depth image is then compared with the camera image using an adapted Structural Similarity Index Measure (SSIM) (Wang et al., 2004), eliminating the need for feature detection, which is challenging due to the different modalities of the sensors.
Once the comparison is made using the SSIM function, the camera-to-IMU transformation is optimized to obtain the new extrinsic calibration parameters. Then the images are compared again to see if the new transform needs to be optimized further or not. With the extrinsic calibration complete, the intrinsic camera calibration follows. For this, a distorted camera image and a depth image made with the same intrinsic parameters, but does not have distortion, are compared in the same way as the extrinsic parameters with the adapted SSIM function to get the focal length and distortion parameters.
The idea of this methodology is to remove the need for specialized calibration targets, pre-calibrated cameras, or a lab environment. However if this methodology is feasible in practice and what its limitations are, is something that needs to be studied.

1.3 Research Questions

To guide this research, the following question is posed:
Main question: Is it feasible to calibrate both the intrinsic camera parameters and the extrinsic parameters between a camera and LiDAR in a multi-sensor system using SSIM to determine how well the resulting images are aligned?
This main question aims to determine if calibration can be achieved by comparing depth and camera images and to identify the challenges that arise from this approach. A sub-research question that was investigated was:

- **What is the margin of error for intrinsic and extrinsic miscalibration?** How accurate are the results from this approach, and are they comparable with traditional or current methods?

1.4 Scope

This thesis focuses on the extrinsic calibration between the LiDAR and the camera, as well as the intrinsic camera calibration. The intrinsic calibration of the LiDAR itself is not covered.

The research is limited to calibrating sensor setups that have one camera and one LiDAR pair. In this research setting the calibration is only done for one pair of the Camera and LiDAR on the multi-sensors system, while it could have more cameras and lidars only one camera is used and one LiDAR. For this multiple systems are tested, the NTU viral dataset with a sensor system on a drone, the Kitti dataset with the sensor system on a car, and the UT-EOS lab's own handheld system.
For the Intrinsics only the focal length and the first radial distortion parameter are estimated, the principal point is assumed to be the center of the image, it is also assumed there is no skew between the XY-axis of the image plane, and the other distortion parameters are set to zero.
It is assumed the initial guess for the extrinsic transform is not too far off from the ground truth value, the same goes for the rotation and focal length these values are between certain bounds, and the initial K1-distortion is set to zero. It is also assumed the error due to motion in the LiDAR depth image is negligible

1.5 Contributions

The primary contribution of this thesis is developing a novel calibration approach that addresses both intrinsic and extrinsic calibration of camera and LiDAR systems, without requiring specialized targets. Unlike other approaches, such as CalibRCNN (Shi et al., 2020) and CalibNet (Iyer et al., 2018), which only determine extrinsic calibration, this method covers both intrinsic and extrinsic parameters. Additionally, the method can be used on prerecorded data of any environment, unlike methods such as (Yan et al., 2023) and (Kümmerle and Kühner, 2020), which require specific setups.
The proposed approach addresses the issue of sensor movement and allows for recalibration when needed, even after data has been recorded. This capability ensures that even minor shifts between the sensors do not make the data unusable, fixing several challenges for existing methods. However, the feasibility of the proposed method needs to be studied, and it is expected this approach does have its limitations.

1.6 Thesis Overview/Outline

The next section covers related work, discussing current approaches and the state-of-the-art for LiDAR-camera calibration. Following that, the methodology section explains what metrics are used, how the LiDAR to IMU transform is determined, and with that transform the pointcloud and Odometry are retrieved. Additionally, this section covers, how the SSIM function is evaluated, which systems and datasets are used for calibration and benchmarking, how the data is preprocessed, and how the intrinsic and extrinsic calibration is done.
The results section presents the outcomes of the SSIM evaluation. Then the results of the extrinsic, intrinsic, and after that a combination of extrinsic and intrinsic calibration are shown. After that, the results are compared of this approach with other models on the same dataset. In the discussion section, the findings and challenges are analyzed, followed by conclusions in the final chapter.

2 Related Work

Understanding the current state of research is necessary to identify gaps and potential areas for further exploration. This section reviews several related papers on the topics of extrinsic calibration between LiDARs and cameras, and intrinsic calibration of cameras, both target-based and targetless approaches.

2.1 Intrinsic Camera Calibration

Intrinsic calibration focuses on determining the camera's internal parameters, commonly referred to as the K matrix, and correcting lens distortion. Traditional methods often involve using targets, such as checkerboards, to accurately compute these parameters. The basic principle is to use a known geometric pattern, detect features in the image, and optimize the camera parameters to make the observed pattern match the known geometry. Three well-known early methods that form the basis of modern calibration techniques are C. Brown's method from 1971, Brown (1971) Tsai's method from 1987 (Tsai, 1987), and Zhang's paper on checkerboard calibration (Zhang, 2000). These works established the framework for target-based calibration, which is still widely used today.

In C. Brown's paper, he introduces a mathematical model to address radial and tangential distortions in camera calibration. Using a Taylor series expansion, he provided a framework for quantifying and correcting these distortions. Radial distortion, characterized by effects like "barrel" or "pincushion" distortion, was modeled using coefficients $k_1, k_2, k_3, \ldots$, which describe incremental deviations from the ideal pinhole projection. Tangential distortion, resulting from lens misalignment, was addressed with coefficients p_1 and p_2, which account for lateral shifts in the image.

Building on concepts like those proposed by Brown, Tsai's method introduced a systematic approach to camera calibration, using a simplified pinhole camera model and refining it by incorporating lens distortion parameters. Zhang's method further improved this process by simplifying the setup, allowing calibration to be performed in less controlled environments while maintaining high accuracy. Both Tsai and Zhang emphasized the importance of capturing multiple checkerboard views from different angles to enhance the precision of intrinsic parameter estimation.

However, despite their accuracy, target-based methods such as checkerboards present challenges. The checkerboards used in calibration are often large and must remain perfectly flat, making them difficult to transport and set up in various locations. In addition, capturing enough images with varying orientations requires significant time and effort. Therefore, targetless methods have emerged as an alternative approach to simplifying intrinsic camera calibration.

Heikkila and Silven (1997) introduced a calibration method based on planar grids and nonlinear distortion modeling, which helped improve the efficiency and accuracy of calibration. Later works, such as li et al. (2008), explored vanishing points for intrinsic calibration, demonstrating that targetless methods could be effective by using specific scene features rather than physical targets.

Targetless methods do not require physical patterns, offering greater flexibility. However, they often require specific conditions or features in the environment. For example, Barreto and Araujo (2005) proposed a method that relies on detecting three or more straight lines in an image to perform calibration, which limits its applicability to specific scenes. More recently, deep learning approaches have emerged as a powerful tool for intrinsic calibration. DeepCalib (Bogdan et al., 2018) uses convolutional neural networks (CNNs) to predict focal length and distortion parameters from input images, offering greater flexibility but at the cost of reduced accuracy compared to traditional methods. While these approaches eliminate the need for physical targets, they introduce challenges related to data quality and

network generalization.

2.2 Extrinsic LiDAR-Camera Calibration

Extrinsic calibration defines the geometric relationship between LiDAR and cameras, which is important for fusing data from these sensors in autonomous systems. Target-based methods remain a popular and reliable choice for doing this, but they require time-consuming setups. Reflective targets like checkerboards or cylindrical objects are placed in the environment, which are visible to both LiDAR and camera sensors. After the calibration process, these targets must be retrieved, further increasing the calibration time.
Levinson and Thrun (2013) gives an early example of using ground truth reflective targets for calibration in autonomous vehicles, being the start of more efficient algorithms. In their approach, highly reflective targets ensured reliable point correspondences between the LiDAR and camera, although the manual setup process remained a challenge.
On the other hand, targetless methods aim to automate the calibration process by identifying natural features seen by both LiDAR and camera frames. Ma et al. (2021) introduced an approach that uses static line features in the environment for calibration, reducing the need for artificial targets. However, recent advances in deep learning have enabled more complex targetless methods that use complex features in the scene.
Neural network-based approaches have become increasingly popular for extrinsic calibration due to their ability to generalize across different environments. CalibNet (Iyer et al., 2018), for example, is a supervised network that predicts extrinsic parameters by minimizing geometric and photometric inconsistencies between LiDAR and camera data. While promising, CalibNet requires accurate camera intrinsics as input, adding an additional layer of complexity to the calibration process. Other networks like CalibRCNN (Shi et al., 2020) and INF (Zhou et al., 2023) have incorporated temporal or density-based features into the calibration process, improving performance, but these methods still have limitations in terms of available training data and generalizability to other environments.

2.3 Combined Intrinsic and Extrinsic Calibration

Some algorithms address both intrinsic and extrinsic calibrations simultaneously, though these methods typically rely on targets. For example, Kümmerle and Kühner (2020) uses a spherical target with ArUco markers to achieve both calibrations and Yan et al. (2023) employs a checkerboard with additional holes for the same purpose.
Recent advancements have explored neural network-based methods that remove the need for targets at all. CaLiCa (Rachman et al., 2023) is one of those examples, where a neural network is trained using LiDAR depth images and corresponding camera images to estimate both intrinsic and extrinsic parameters. Unlike target-based methods, CaLiCa eliminates the need for physical objects in the environment, relying instead on image-based features. However, the key difference of our approach lies in using multiple LiDAR frames to improve depth image quality so a better comparison between the depth image and camera image can be done. The code to CaLiCaNet is not open source or available it could not be tested. By using more dense data from multiple scans, a higher level of precision is aimed to achieve in both intrinsic and extrinsic calibration compared to single-frame methods.

2.4 Pointcloud and Odometry Generation

To generate depth images from LiDAR frames, a pointcloud must first be created from the LiDAR data. This step is important for producing accurate depth images that can be compared to camera images, increasing the overall calibration process. Additionally, the

pose of the camera relative to the LiDAR must be known, to make the depth images at or near the camera pose. While the exact LiDAR-camera transform is the primary goal of the model to determine, the LiDAR-IMU-Init package Zhu et al. (2022a) is used to estimate the LiDAR-IMU transform. With an initial guess for the IMU-camera transform, depth images can be generated. The pointcloud and LiDAR odometry are produced using Fast-LIO-SLAM Kim (2024a), an optimized version of Fast-LIO enhanced with SC-PGO.

2.4.1 Fast-LIO

Fast-LIO Xu and Zhang (2021), or Fast LiDAR Inertial Odometry, is a ROS package that integrates IMU data and LiDAR messages to determine the odometry. As input, it requires IMU data, LiDAR messages, and the estimated transform between the LiDAR and IMU. This transformation can be determined using another ROS package, LiDAR_IMU_init, which utilizes the motion of both sensors to compute the transformation. Although Fast-LIO generates a pointcloud and odometry, these outputs may contain errors. These errors can be reduced through, loop closure and pose-graph optimization with SC-PGO.

2.4.2 SC-PGO

The SC-PGO ROS package Kim (2024b) utilizes the poses and motion-compensated LiDAR frames generated by Fast-LIO. It uses loop closure and pose-graph optimization to optimize the poses, resulting in a more accurate fusion of the LiDAR frames and a higher-quality pointcloud. Loop closure is done by identifying features in LiDAR frames and recognizing previously visited locations based on these features. The algorithm tries to establish connections between all frames, and an optimization algorithm is employed to refine the poses. Each pose generates a local pointcloud, which can be merged into a larger pointcloud using an additional provided Python script. After analyzing the merged pointcloud, some erroneous points were identified and filtered out. The filtering process involved examining the density of points within a sphere of radius 0.5m; points with insufficient neighbors were removed. Following these steps, a high-quality pointcloud was produced, ready for depth image generation.

3 Methodology

In this section, the method to estimate the intrinsic and extrinsic parameters between the LiDAR and camera pairs is presented. First, the overall approach is outlined, including the steps required to perform both the extrinsic and intrinsic calibration. Then, the different datasets used for testing the calibration are described, specifying the sensors involved and the locations where the data was recorded. Next, the method for creating depth images is discussed. Following this, the calibration process is detailed, starting with the SSIM function—how it operates, how it is adapted for this method, and how it is analyzed for changes in extrinsic and intrinsic values. The SSIM function is then utilized for intrinsic, extrinsic, and combined calibration of extrinsic and intrinsic parameters.

3.1 Approach

To calibrate the multi-sensor setup several steps are done after each other. But the main steps to determine the calibration are: gather data, preprocess the data, determine the extrinsic parameters, and determine the intrinsic parameters. The data used for testing the calibration are gathered from the internet, the Nanyang Technological University (NTU) from Singapore has online datasets of drones with multiple lidars and cameras Nguyen et al. (2022). The Kitti dataset Geiger et al. (2012), is another that can be used, and it is often used by other papers focused on calibration. But Kitti is made on public streets with moving objects that can affect the calibration. So the NTU dataset is used to test, because of minimal moving objects in the data and the ground truth values. Besides the public dataset, data is also gathered with a handheld sensor setup of the group, and the data is recorded in the "Langezijde" building at the university, this is one of the original sensor setups that this project was aimed at. With this also the checkerboard calibration can be done and the traditional approach and the approach developed in this thesis can be compared.
The LiDAR data is used to make a pointcloud and determine the trajectory of the sensor system, with the use of Fast-LIO-SLAM. With the pointcloud and camera images the extrinsic calibration is done through an iterative approach. With the extrinsic calibration done the intrinsic calibration can be done, this is done the same way as the extrinsic calibration but only the Focal length and K1 distortion parameter instead of the transform are optimized. In the traditional approaches first the intrinsics are determined and after that the extrinsic but for the intrinsic the straight lines, edges, or objects in the depth image need to be compared to the same lines, edges, and objects in the camera image, for this first the extrinsic need to be determined and then the intrinsic. While it may be possible to combine the calibrations into one simultaneous calibration this makes the search space for the optimizer bigger and possibly too big and together with different step sizes and terminal tolerances this is not done. This approach is visualized in Figure 2.

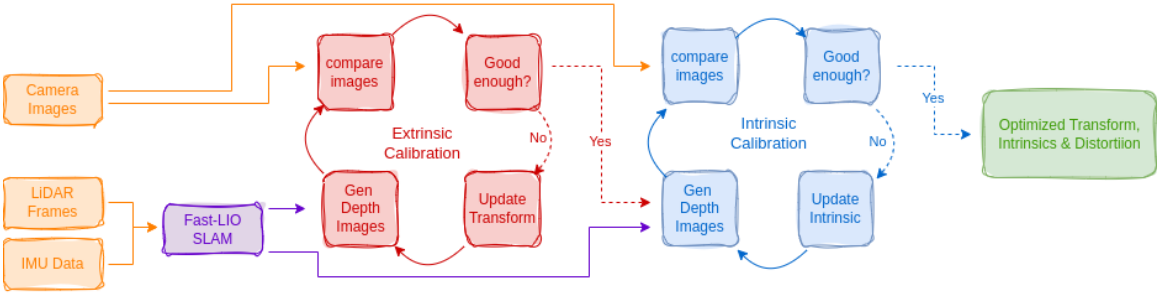

Figure 2: Flow chart of the calibration Approach

3.2 Datasets

To train and evaluate the model several datasets are used, the first dataset is gathered with our sensor setups, the Handheld combination with one Camera, one IMU, and one LiDAR mounted on a 3D printed mount. The second dataset is the NTU dataset which has 2 cameras, 2 lidars, an IMU, and some other not-used sensors, these are all mounted on a Drone, and the footage is taken from an urban environment. The third Dataset is the Kitti dataset, which is used for benchmarking, the data is gathered with 2 cameras, 1 lidar, and one IMU all mounted on a car (Volkswagen Passat B6). Each of these datasets differs in terms of the number and configuration of sensors, are taken in different environments, and are mounted differently which will result in different movement, stability, and noise. In the next subsections, the different sensor setups are explained in more detail, including which sensors are used at which measurement frequency and what kind of environment.

3.2.1 Handheld Combination

The Handheld combinations use a Realsense D435i Intel Realsense (2019) camera which takes an image at 30Hz, with a resolution of 480x640. The LiDAR is the Hesai pandar XT32 LiDAR Hesai Technology (2021) which takes a scan at 10Hz and as the names suggest has 32 beams. The IMU is the XSensMTI630R IMU XSens (2024) which measures at 400Hz and measures the xyz-acceleration and rotation around the xyz axis. While the camera can measure also depth, this is not used since the goal of the model is to compare depth images made from the lidar not from the camera. The sensors are connected to a 3d printed handle that is made from plastic, the lidar is on top, the IMU is to one of the sides of the handle, and the Camera is on the other side of the handle compared to the IMU, a photo of the combination can be seen in figure 3. The data gathered by this setup is of the inside of the Langezijde of the University of Twente, the building has 2 levels and contains areas of vegetation, many offices, hallways, Windows, and a big stairway/seating area. Example images can be seen in figure 24 in appendix A. There can be seen images of the staircase/seating area, the shorter hallways with areas of vegetation, and the long hallway going through the whole building also with areas with vegetation.

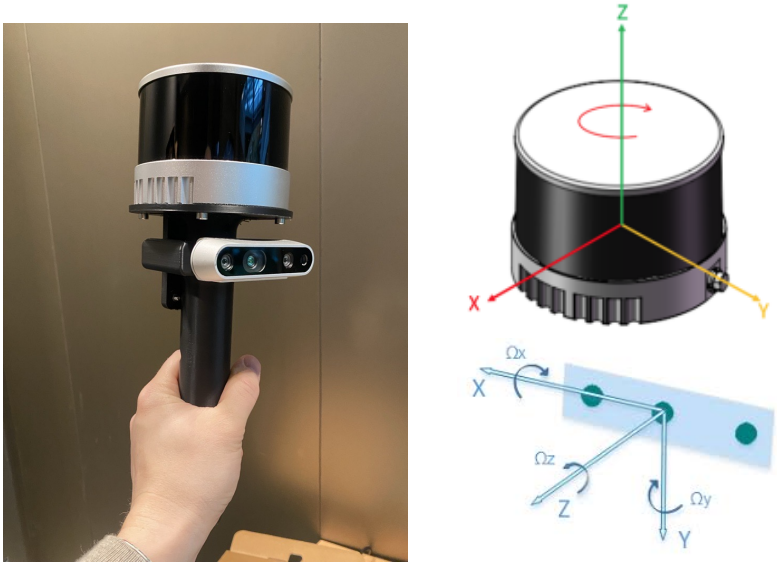

Figure 3: Photo of the handheld combination, with the axis for the Hesai Pandar XT-32 LiDAR and Realsense D435i camera.

3.2.2 NTU Dataset

The NTU VIRAL(A Visual-Inertial-Ranging-Lidar Dataset for Autonomous Aerial Vehicle) dataset, is a dataset made available by the Nanyang Technological University of Singapore. There are twenty datasets, six locations measured three times, and then 2 calibration datasets. The sensor setup is mounted onto a drone, and the two cameras that are used are the uEye 1221 LE with a measurement frequency of 10Hz. The LiDARs are two Ouster OS1-16 gen 1, with 16 beams and a measurement frequency of also 10hz. The IMU is the VectorNav VN100, which measures 385 times per second, in figure 4 the sensor setup of the drone is shown. What makes this public dataset useful is that it has well-known extrinsic and intrinsic parameters between the different sensors, which can be used to determine the error in the estimation and it has a lot of data for the model to be trained on. In figures 25 and 26 in appendix A, images of the eee_01 and sbs_01 datasets can be seen. In the NTU eee_01 dataset, the environment has on 3 sides buildings with a repetitive pattern per floor of the building, and on the 4th side a green space with a tree and some other vegetation. In the sbs_01 a similar environment is shown but with more glass, less similar buildings, and more vegetation.

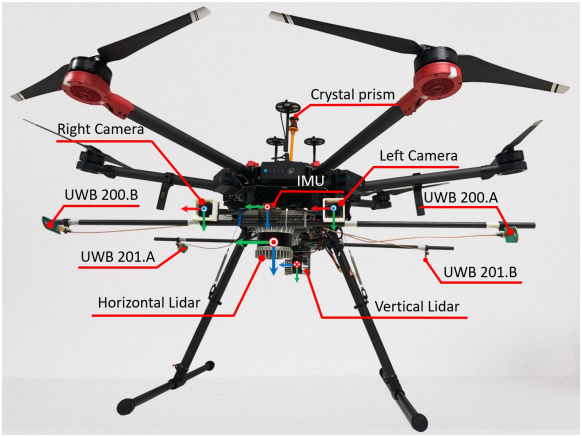

Figure 4: NTU Viral Drone Sensor system

3.2.3 KITTI Dataset

The Kitti Odometry dataset is a dataset made for advancements in autonomous driving technology, this dataset is used in many other papers to train their neural network. The Kitti Odometry dataset is measured with Volkswagen Passat B6 in the German city of Karlsruhe. The Cameras used are the Point Grey Flea 2 (FL2-14S3C-C), with a measurement frequency of 10Hz. The images are wide-angle images and produce images with a resolution of 1242x375, this is much wider than the images from the handheld or the NTU dataset. The LiDAR that is used in the Velodyne HDL-64E, has 64 beams and also measures at 10Hz. The IMU used is the OXTS RT 3003 which measures at a frequency of 100Hz, which is lower than all the other IMU but since it is mounted onto a car the possible movements are more limited than for example a drone, which means a lower frequency may not give a worse Odometry estimation. How these sensors are mounted can be seen on their website or in figure 5. This dataset has the same advantage as the NTU VIRAL dataset, a lot of data and well-known ground truth, but is also used by other papers which means our results can be used to compare. The NTU dataset was mostly used because of the environment it was recorded in, which has minimal objects/humans in it that move. In figure 27 in appendix A, example images of the Kitti dataset can be seen, these images have a different size than the images of the other datasets, these are wider. In the cars, vegetation can be seen.

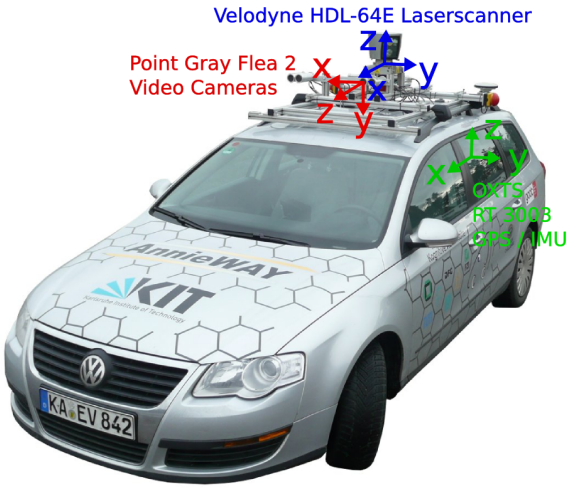

Figure 5: The Kitti sensors setup

3.3 Depth Image Generation

To make the depth image for its associated camera image pair, the pose of the camera image is needed, as are the intrinsic camera parameters, and the pointcloud of the environment. A flow chart is shown in figure 6 of how the depth image is made. The Pointcloud and the Odometry are determined with the Fast-LIO-SLAM with the LiDAR and IMU data. Since the Odometry is the pose of the body frame (which is where the frame of the IMU), to make the depth images in the same pose of the camera the transform between the Camera and IMU is needed, which is optimized for this thesis. Together with the transform between the LiDAR and the IMU, the LiDAR to Camera transform can be determined.
First, the pose of the IMU is determined at the time when the image was taken, this is done by making a spline between odometry poses and the time of that pose, then using the image time to get the Pose. After that, the pose of the camera is determined by the transform between the IMU and the camera. Then with the pointcloud, pose, and intrinsic parameters, the depth image is made with the use of open3d. This makes depth images with the same size and position as the camera image, but the pixel value is the depth instead of color. For the SSIM comparison, the depth value should be normalized to the same values as the camera image pixel values, which is between 0 and 255. To determine the pixel location in the 2D camera image for a given 3D point in the pointcloud, the point's coordinates, the transformation matrix T, and the camera intrinsic matrix K are required, as shown in Equation 1. This equation is used to determine the location of a 3D point (in the pointcloud) in the 2D depth image. The Lidar on the NTU drone has a maximum range of 200 meters, the Hesai Pandar XT32 and the Velodyne off the Kitti measure up to 120 meters. To be below 120 meters, the max depth is set at 100 meters, and all values below that are clipped to this value, these are then converted to values between 0 and 255.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K * T * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

3.4 Handheld Combination Checkerboard calibration

The handheld sensor setup does not have a predefined ground truth, unlike datasets such as NTU or KITTI, as this is a custom-built sensor configuration provided by the supervi-

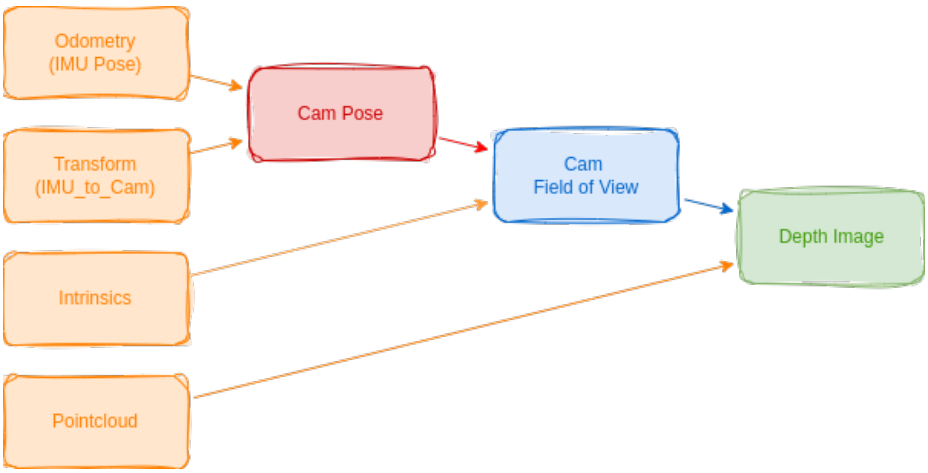

Figure 6: Flowchart og how depth images are made

sor. Consequently, a ground truth must be determined to evaluate the performance of the automatic targetless calibration approach this project is about. This ground truth calibration is performed using a checkerboard displayed on a screen or monitor, which serves as a good alternative when a physical checkerboard is unavailable. Screens are everywhere and provide the flat surface necessary for calibration.
To do the calibration, data must first be recorded. The screen should be captured from different angles and positions to ensure accurate calibration. For intrinsic calibration, the checkerboard pattern is detected in as many images as possible. These detections are then processed using the OpenCV camera calibration function to calculate the intrinsic matrix (K-matrix) and distortion coefficients.
Extrinsic calibration is more difficult, as it needs to identify corresponding points in the camera image and the LiDAR frame. In the camera image, checkerboard corners can be used as reference points. For the LiDAR frame, the corresponding plane on which the checkerboard is displayed must be identified, given its dimensions are known. This involves:

- Segmenting the LiDAR frame to identify planes.
- Selecting the plane that matches the screen.
- Determine the sides of the plane.
- Determine the corners, where the sides of the plane intersect.
- Adjusting for the housing around the screen to accurately locate the checkerboard edges.

Once corresponding corners are identified in both modalities, additional points can be derived. From the four detected corners, a grid of points (with x rows and y columns) is generated in both the camera image and the LiDAR frame. These points, along with the previously determined intrinsic parameters and distortion coefficients, are used in the OpenCV solvePnP function to estimate the rotation and translation (extrinsic).
To enhance the accuracy of both intrinsic and extrinsic calibration, it is essential to detect the checkerboard and its corresponding plane in as many images and frames as possible.

3.5 Calibration

In this work, an Adapted SSIM function for both intrinsic and extrinsic is used for the calibration processes. Before detailing the specific calibration techniques, an overview of

the Adapted SSIM function is provided, explaining its modifications and how it contributes to the calibration process.

3.5.1 Structural Similairity Index Measurement (SSIM)

Both calibrating the extrinsic and intrinsic parameters is done by comparing depth images to camera images. This is through SSIM which stands for structural similarity index measurement, which returns a value for how similar two images are. With a value of 1 for totally similar, 0 for not similar, and a value of -1 for totally anti-similar. SSIM consists of comparing 3 parts, the luminance, the contrast, and the structure of the image, with a weighting between the 3 parts. The 3 parts are as follows:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{2}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{3}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{4}$$

with $c_3 = \frac{c_2}{2}$.

Where:

- μ_x is the mean of x,
- μ_y is the mean of y,
- σ_x^2 is the variance of x,
- σ_y^2 is the variance of y,
- σ_{xy} is the covariance of x and y,
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$, and $c_3 = \frac{c_2}{2}$ are constants used to stabilize the division.

The final SSIM index is a weighted combination of these comparisons:

$$\text{SSIM}(x,y) = l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma$$

If these weights are set to 1, SSIM reduces to the following formula:

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5}$$

So the luminance compares the mean of the pixel value, the contrast compares the variance between the pixel values, and the structure uses the covariance between the pixel values of both images. Since the camera and depth images are made with different modalities, the luminance value is not applicable here and α is set to zero, which results in $l(x,y)^\alpha$ always being one.

$$\text{SSIM}(x,y) = 1 * c(x,y) * s(x,y) \tag{6}$$

What do the SSIM value and the individual values for the structure and contrast components signify, especially considering that the pixel values in the two images represent different meanings? In a depth image, the pixel values indicate the distance of a point from the camera's pose when the depth image was captured. If the depth is normalized, the

maximum pixel value (either 255 or 1) represents the maximum depth chosen, while a value of 0 indicates a distance of zero.
In contrast, the pixel values in a camera image represent intensity as measured by each color sensor, where a pixel value of 0 corresponds to black and a value of 255 (or 1) corresponds to white in a grayscale image. This creates a situation where the interpretations of black-and-white images are essentially opposite: a point that is far away and the camera sensors receive little light from that point will have a low pixel value in the camera image but a high pixel value in the depth image, and vice versa. Therefore, it is expected that a lower SSIM value indicates a better comparison between the images.
SSIM is calculated for a window of NxN, by default "skimage" uses a window size of 7. This window slides over the images and then computes the SSIM for each window, the window slides by 1 pixel every time, to get the final SSIM value the mean is taken over all the windows. For our application, bigger window sizes are used to get the matching structure.

SSIM Eval method

We study the feasibility possible to use SSIM to compare the depth images and the camera images, and how precise the extrinsic and intrinsic parameters can be determined, the SSIM needs to be evaluated. This is done by slightly adjusting one of the parameters from the Ground truth and then determining the SSIM value. This is done for all 6 degrees of freedom (X, Y, Z, roll, pitch, yaw) and then also for the intrinsic parameters (Focal length and K1 distortion parameter) and for different window sizes.

3.5.2 Extrinsic Calibration

Our approach does not work by matching the features, but by comparing the camera image with the depth image. This is done by changing the Transformation between the IMU and the Camera and seeing how the SSIM (structural similarity index measurement) changes. The optimization is done with the optimize-minimize function from ScipyVirtanen et al. (2020), and several variables can be changed to do the best optimization. This function needs an objective function to minimize, this function needs to return a single value based on the input that needs to be optimized. In this approach, this input is the IMU to Camera transform and the output is the SSIM value. To be less affected by a random bad camera or depth image, the average SSIM value is taken over several images.
The optimization uses the "L-BFGS-B" algorithm because it is good for high-dimensional optimization problems because of its efficient memory use, relying on a limited-memory version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. Additionally, with "L-BFGS-B" constraints on variables can be bounded, allowing for control over parameter limits, which is useful when optimizing parameters with physical constraints and an exproximate error in the initial guess. Also, as a gradient-based method, it convergence faster and gives higher precision in smooth optimization landscapes, which is good for minimizing complex, continuous objective functions.
Now that the solver is chosen, some parameters can be set, for example, how many maximum iterations or function evaluation, or when is the optimization good enough and how big should the step size be. Many solvers have their special solver options, sometimes the default is good bit also often the values need to be determined with trial and error. Therefore, the optimization process should be designed to stop within a reasonable timeframe, ensuring it does not run forever while still allowing enough iterations to achieve accurate calibration. For the function to know when to stop when the optimization is good enough a parameter can be set called Ftol, when the SSIM value does not improve anymore and it is

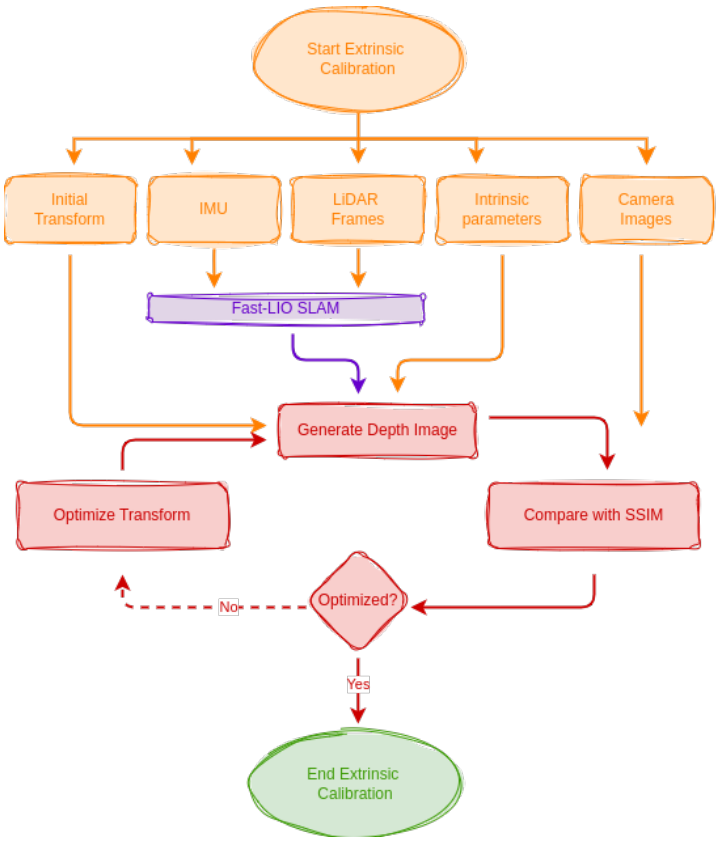

Figure 7: Flowchart of the Extrinsic calibration

below Ftol value it will terminate. Ftol can be determined with the SSIM evaluation plots and the wanted maximum error, but since the extrinsic calibration won't be done with 1 parameter but with 6 the precise value is hard to determine. How the optimization works are shown in figure 7, start with a good initial guess, determine SSIM, and Optimize the transform till it is well optimized.
After the calibration is done on rectified camera images, the calibration can be tested on the distorted camera image from the camera. If the extrinsic calibration can be done with distorted camera images the combination of extrinsic and intrinsic will be easier to do.

3.5.2.1 Loss metric

To see how good the result of the optimization is, the difference between the result and the ground truth needs to be determined. For the translation, this is the Euclidean distance between the two poses. For rotations, multiple sets of Roll, Pitch, and Yaw values can represent the same final rotation, which introduces some complexity in comparisons. Since rotations are optimized using quaternions, a quaternion distance function is employed to quantify the difference between rotations.
The quaternion distance function calculates the difference between two rotations in quaternions by first normalizing both quaternions, then calculating the absolute dot product and calculating the arcos or inverse cosine to get the difference in radians. This is then converted to degrees to get the loss.

3.5.2.2 Calibration Evaluation

To see how well this approach works and which parameters work, some tests need to be done. To evaluate the extrinsic calibration approach, the ground truth transform is adjusted

with a random transform between bounds, then the optimizer is run. This needs to be done several times, at least 10 but more is better. Then the results need to be analyzed to see what the average error between optimization results and how much it deviates between runs.
After it works with undistorted images, it can be checked how well the extrinsic calibration works with distorted images.

3.5.3 Intrinsic Camera Calibration

The pinhole model is foundational in understanding camera internal parameters. Under this model, the camera's focal length and principal point are represented as parameters that define how points in 3D space project onto the camera's 2D image plane. Specifically, focal length controls the scaling of the projected image, effectively determining how zoomed in the image is, while the principal point sets the image center where the camera's axis intersects the image plane. In equation 7, the K-matrix is stated, containing five variables: the focal lengths in the X and Y directions, the X and Y components of the principal point, and the skew term between the X and Y axes of the image plane. However, the skew term is assumed to be zero due to the quality of modern cameras. The pinhole model assumes an idealized, distortion-free camera, simplifying the geometry and making it suitable for calibrating cameras with minor lens distortion. This model allows us to form the intrinsic camera matrix, which encodes the internal geometry that must be known or estimated accurately for tasks like 3D reconstruction or sensor fusion.

$$\mathbf{K} = \begin{bmatrix} f_x & S & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Image distortion describes how certain optical imperfections in the camera lens cause images to deviate from an ideal, undistorted projection. Two common types of distortion are barrel and pincushion distortion. Barrel distortion causes the image to appear to "bulge" outward at the center, creating a convex effect, while pincushion distortion makes the image edges appear to pinch inward, producing a concave effect. These distortions can significantly affect the accuracy of measurements and calibration if uncorrected, as they alter the geometry of captured images.
In calibration, distortion is modeled using parameters that quantify the deviation from the ideal pinhole projection model. The radial distortion parameters, typically labeled as k1,k2, and k3, correct for the bulging or pinching effects by adjusting the radial displacement of each pixel relative to the image center. These parameters progressively adjust for larger distortions that occur further from the image center, with k1 often having the greatest influence, followed by k2 and k3 for finer corrections. Tangential distortion parameters, labeled p1 and p2, account for asymmetries in the lens alignment, correcting lateral shifts that cause the image to appear tilted. In equation 8 the way the radial distortion is modeled, and in equation 9 how the tangential distortion is modded, where $r^2 = x_\mathrm{u}^2 + y_\mathrm{u}^2$. Here $x_\mathrm{u}, y_\mathrm{u}$ are the pixel location of the undistorted image and, $x_\mathrm{d}, y_\mathrm{d}$ the pixel location in the distorted image. Since only the K1 parameter is estimated only the radial distortion is needed and only the first part of it.

$$\begin{aligned} x_\mathrm{d} &= x_\mathrm{u}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots\right), \\ y_\mathrm{d} &= y_\mathrm{u}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots\right) \end{aligned} \tag{8}$$

$$\begin{aligned} x_\mathrm{d} &= x_\mathrm{u} + \left[2p_1 xy + p_2(r^2 + 2x^2)\right], \\ y_\mathrm{d} &= y_\mathrm{u} + \left[p_1(r^2 + 2y^2) + 2p_2 xy\right] \end{aligned} \tag{9}$$

In the calibration workflow, these distortion parameters are optimized to minimize the error between observed and expected points in the image. Properly determining and correcting these distortions is important to accurately map pixel coordinates to real-world coordinates, resulting in better precision for tasks such as sensor fusion, 3D reconstruction, and image rectification.
To calibrate the Intrinsic parameters for the camera, a similar method is used as in extrinsic calibration. But instead of optimizing the transform, the intrinsic parameters are optimized, these are the Focal length and the K1 distortion parameter. To optimize the focal length, the camera image stays unchanged, but the depth image is made with different focal lengths to see which focal length is the best one.
To calibrate the distortion the distortion value can be optimized, then undistort the image with this value and then use SSIM to compare it to the depth image. The problem may be to augment the image so that it will have a black border around the image if the distortion is barrel distortion. Since there is then a black border the SSIM value will be less. In figure 8 the intrinsic calibration flow chart is shown, the image is similar to the extrinsic calibration flowchart, because the calibrations are similar. The depth image is made with the ground truth extrinsic transform, Odometry, Pointcloud, and the optimized focal length. The camera image is undistorted with the optimized focal length and optimized distortion parameter. These are then compared with SSIM and see if the value is good enough or does not change anymore.

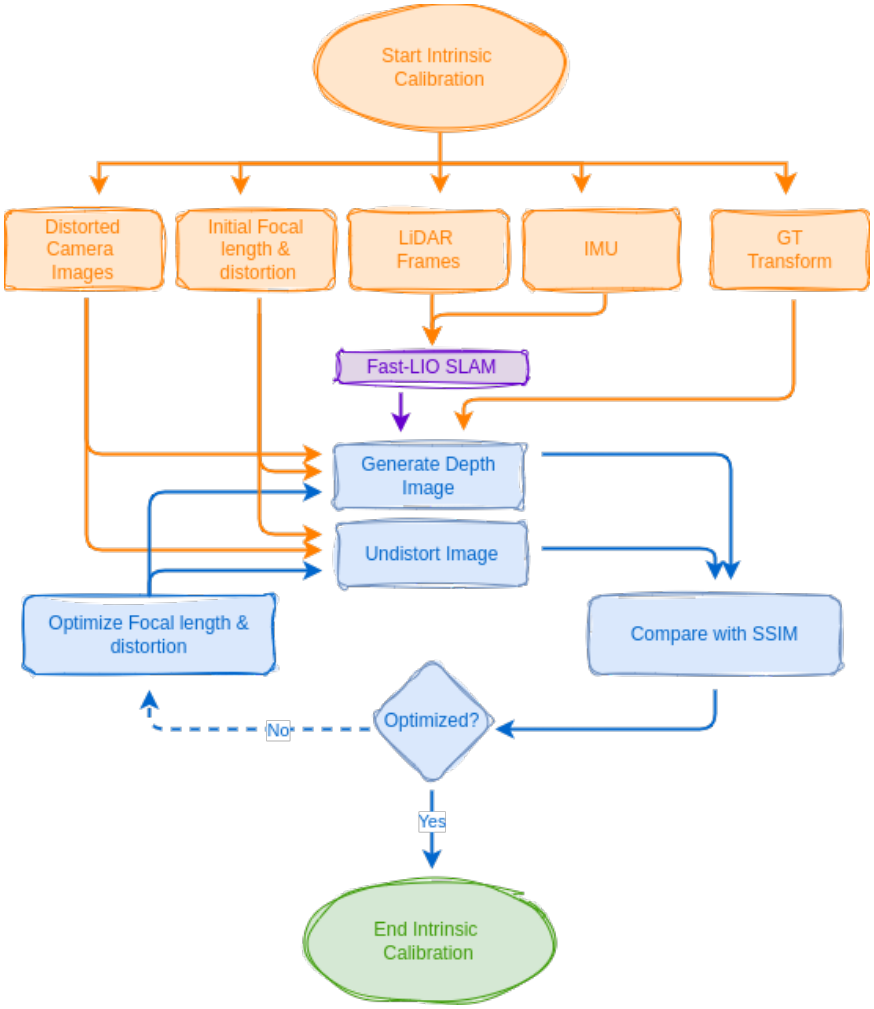

Figure 8: Flowchart of the Intrinsic calibration

3.5.4 Intrinsic and Extrinsic Calibration Combination

When both intrinsic and extrinsic calibrations perform well separately, they can be combined into a joint optimization process. This approach allows for simultaneous optimization of both Extrinsic and intrinsic calibration but also increases the search space, making the process more computationally complex. To reduce this, calibration is conducted in stages: performing extrinsic calibration first on distorted camera images, followed by intrinsic calibration. This approach is visualized in the flowchart in figure 2 at the beginning of the methodology.
The primary goal is to estimate translation, rotation, focal length, and distortion. If certain parameters are found to be difficult to estimate accurately, calibration can be adjusted to focus on a subset of the parameters for improved results.
Initial calibration will be conducted using the NTU dataset, while full calibration will be tested on an additional dataset to assess performance on other datasets. Additionally, results will be compared to other papers that use the KITTI dataset to evaluate performance against existing methods.

4 Results

In this section the results of the feasibility of using SSIM in extrinsic and extrinsic calibration. First, the SSIM function is evaluated by seeing the effect on the SSIM value when one degree of freedom is adjusted from the ground truth value. These adjustments will be for the translation (X, Y, Z), Rotation (Roll, Pitch, Yaw), Focal length, and K1-Distortion parameter. The calibration tests are done, first for the extrinsic and intrinsic separately and then in a combined approach. These results will then be discussed in the discussion.

4.1 Checkerboard Calibration

For the Kitti and NTU datasets, the ground truth intrinsic and extrinsic are known but for the Handheld combination, this is not known and needs to be determined with a different method. This is done with a checkerboard calibration, where the checkerboard is displayed on a screen and recorded by the Handheld LiDAR and Camera combination. First, the intrinsic Camera parameters will be determined, and after that the extrinsic translation and rotation between the LiDAR and Camera.

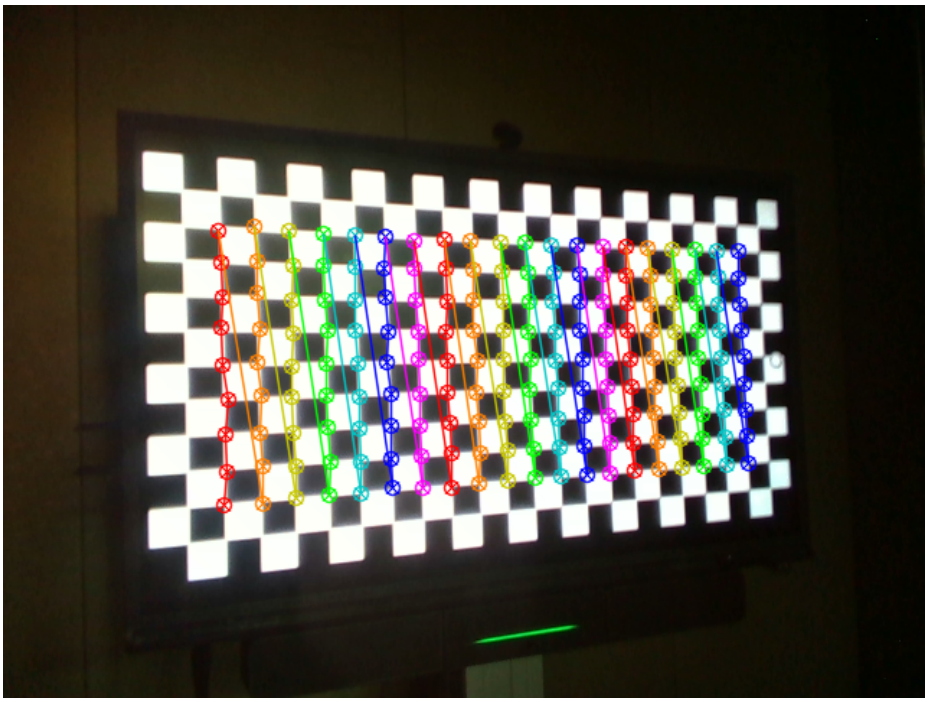

Figure 9: Camera image with detected checkerboard

4.1.1 Intrinsic Calibration

To determine the intrinsic calibration parameter values of the camera a checkerboard calibration method is used. Camera images are recorded of a checkerboard pattern on a computer monitor. The data recorded consisted of a screen displaying a checkerboard pattern. Images where the checkerboard was successfully detected were identified. Since the checkerboard was displayed on a screen and with the varying lighting conditions in the room, the detection was not in every image. However, the checkerboard was detected in more than 300 images out of the total of 4800 images in the 3 bags. In figure 9 a found checkerboard is shown. In the checkerboard there is a problem, the algorithm normally finds the inside corners of the checkerboard with one square border. In the figure it could be seen there is a double square border, however, in all the images there is a double border

and the same inner corners are found. The real-size checkerboard, with a one-square border, could not be found in any of the images.
Three Rosbags were used that contain images from many different angles and positions with the checkerboard at least once in each corner and in the middle, this is good for determining the (radial) distortion. After the corners of the checkerboard are found it is given to the "calibrateCamera" function of OpenCV together with the physical distance between points. The results can be seen in equation 10, comparing it to the intrinsic parameters from what is in the camera info message in equation 11, it can be seen that the values are not very different from each other. After determining the intrinsics, the reprojection error was calculated, with a mean value of 0.0769. The reprojection error represents the distance between the actual image points and the reprojected points, computed using the estimated intrinsics. On average, the reprojected points were 0.0769 pixels away from their actual locations. K1 can then be used in equation 8 to rectify the image together with the K matrix. Here, only the first distortion parameter (K1) is estimated. The other parameters are not estimated due to the limited number of images and the targetless approach, which focuses solely on estimating the first distortion parameter.

$$K = \begin{bmatrix} 617.2 & 0 & 327.6 \\ 0 & 616.3 & 238.9 \\ 0 & 0 & 1 \end{bmatrix}, Distortion = [0.1179, 0, 0, 0, 0] \tag{10}$$

$$K = \begin{bmatrix} 611.8 & 0 & 326.8 \\ 0 & 611.8 & 240.6 \\ 0 & 0 & 1 \end{bmatrix}, Distortion = [0, 0, 0, 0, 0] \tag{11}$$

4.1.2 Initial values for extrinsic calibration

Extrinsic calibration can be performed using an automatic approach. However, to assess the accuracy of the results, the initial extrinsic transforms are manually determined. In equations 12,13, and 14 the homogenous transforms are stated. Here it was assumed all the sensors were at right angles from each other and the distances between the sensors were measured with a ruler.

$$T\_LiDAR\_IMU\_Derivation = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0.08 \\ 0 & 1 & 0 & -0.05 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$T\_IMU\_Cam\_Derivation = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$T\_LiDAR\_Cam\_Derivation = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -0.08 \\ 0 & -1 & 0 & -0.05 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

4.1.3 Extrinsics Autocalibration

The extrinsic calibration between the camera and LiDAR is also done by an automatic approach. The approach detects the checkerboard in the camera image and then detects the plane of the screen where the checkerboard is shown. From there points are matched and together with the intrinsic parameters the extrinsic calibration is done using

the "cv2.solvePnP" function. The points are matched for as many images as possible to get an accurate as possible calibration. The resulting transformation can be seen in equation 15, the rotation matrix seems similar to the manually determined approach but the translation is much bigger. For this, 180 points are matched between the two modalities in a grid of 20 by 9, for 300 pairs of images and point clouds.

$$T\_LiDAR\_Cam\_Automatic = \begin{bmatrix} -0.99 & 0.14 & 0.02 & 0.36 \\ -0.02 & 0.05 & -1.00 & -0.08 \\ -0.14 & -0.99 & -0.05 & 2.86 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{15}$$

4.1.4 Matlab lidarCameraCalibrator

The third approach, besides the manual derivation and the Python OpenCV solvep2p method, was using Matlab Mathworks. Matlab provides a lidarCameraCalibrator app that loads camera images and LiDAR data to estimate the extrinsic calibration between the two sensors. This method was explored after the auto-calibration approach was not good enough, and gave a significant translation error.
In the lidarCameraCalibrator app, several inputs were required, including the size of the checkerboard, the padding of the housing, intrinsic camera parameters, and an initial extrinsic guess. While the calibration process could load the data, it was unable to automatically detect the checkerboard in the point cloud. Manual annotation was necessary for all 67 checkerboards, which was time-consuming.
Once the annotations were complete, the calibration process provided the results shown in Equation 16. The rotation estimation was comparable to both the manually derived and auto-calibration transforms, but the translation was more realistic and aligned better with the manually derived result. After projecting the LiDAR points onto the camera image and calculating the reprojection error, an average error of 11.68 pixels was determined.

$$T\_LiDAR\_Cam\_Matlab = \begin{bmatrix} -0.99 & -0.01 & 0.02 & 0.0084 \\ -0.02 & 0.01 & -0.99 & -0.0277 \\ 0.01 & -0.99 & -0.01 & -0.0196 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{16}$$

4.1.5 Comparison

For the handheld combination, three different transforms were determined. To evaluate which transform is the most accurate, they were used to generate depth images. These depth images are shown in Figure 10. Upon inspection, the depth image generated using the automatic transform appears to align with the camera image the least. In contrast, the images produced using the manually derived transform and the Matlab estimation show a better alignment with the camera image, but because there is only a small difference between the two transforms this won't be visible in the depth images.

4.2 SSIM feasibility in calibration

To evaluate the feasibility of using SSIM to determine the similarity between camera and depth images, the effect of adjusting one of the calibration parameters on the SSIM output is determined. The SSIM value was calculated for the six transformation parameters of the extrinsic parameters, along with the focal length and the K1 distortion coefficient of the intrinsic parameters. These parameters were varied with predefined ranges, the translation was adjusted between ± 2 meters, the rotation between ± 90 degrees, the focal length between ± 50 pixels, and the K1 distortion coefficient between ± 1.0. For each parameter, 101

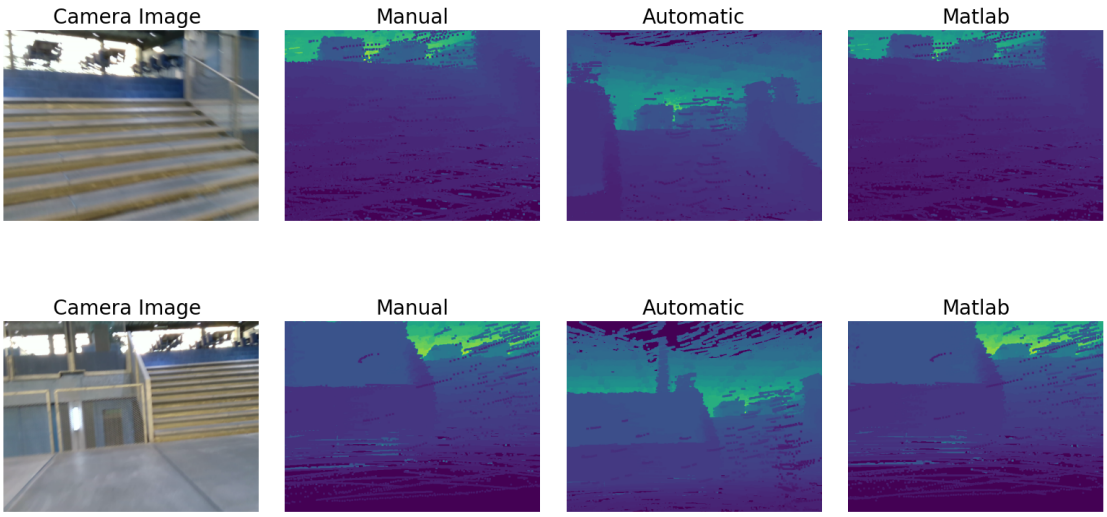

Figure 10: Camera and Depth images of the handheld combination made with the three determined transforms.

samples were used within these ranges. The test is done a second time with a tenth of the first set of ranges, for the same set of images and also for 101 samples to get a finer detail around an adjustment of 0.
To determine SSIM's sensitivity to different adjustments, multiple window sizes were used for SSIM calculation. These window sizes were selected as one less as a power of two, ranging from 7 to 255, to see which size is the best for which calibration. Furthermore, the SSIM values were averaged over 50 images from the dataset, providing a reliable estimate while lowering the impact of possible outliers or random noise in some images.
In figs. 11 to 13 the SSIM evaluations for the translational adjustments along the X, Y, and Z axes are shown. In these plots, the image coordinate axes of the depth and camera images are parallel, with only translational adjustments applied along each axis and all other values are the ground truth value. For each translational direction, there are two plots: one showing results for large adjustment bounds, and another for smaller bounds, a tenth of the bigger bounds, centered around an adjustment of zero. The minimum SSIM values for each window size are highlighted with a dot to highlight the peak value, and to which the optimizer would optimize. When using large bounds, distinct SSIM peaks can be seen for larger window sizes. However, these peaks level out as the adjustment bounds become narrower. The peaks are also not (perfectly) near the zero adjustments.
For the rotational components of the extrinsic transform, shown in figs. 14 to 16, SSIM was evaluated by rotating the depth image along each of the Roll, Pitch, and Yaw axes while keeping the image origin fixed. As with translation, two plots were generated per axis, representing large, and narrow bounds for the rotation adjustment. For the translation, the peaks leveled out and could not be seen in the right plots, but this was not the case for the Rotational SSIM evaluation. The rotational adjustments SSIM Evaluation graphs maintained visible SSIM peaks even for narrower bounds, with these peaks aligning closer to zero adjustments. Additionally, as with the translation parameters, larger window sizes resulted in lower SSIM peaks. This indicates an increase in sensitivity for larger window sizes when dealing with rotational adjustments. However, the biggest window size is not the best as can be seen in the Pitch adjustment plot (figure 15) the biggest window size has a lot of peaks and valleys. This means that if the initial guess for the roll axis has a large error, the optimization may converge to an incorrect minimum SSIM value.

For focal length adjustments, Figure 17 shows the SSIM evaluation across different window sizes. In this evaluation, depth images were generated with varying focal lengths, while the extrinsic pose between the depth and camera images remained at the ground truth value. In the plot for the focal length, the peaks are not well defined, even when looking at the left plot with bigger bounds. This highlights a limitation of this methodology
Finally, Figure 18 shows the SSIM evaluation for the K1 distortion parameter across various window sizes, using raw images from the NTU eee_01 dataset. In this case, only the K1 distortion parameter was adjusted, with each image undistorted according to the adjusted value before SSIM calculation. As observed with other parameters, larger window sizes gave lower SSIM values. Here a similar thing for the translation evaluation can be seen, in the plot with big bounds there is a peak but in the right plot with small bounds, the peak flattens out. The peak value is also a lot more the the right, with the ground truth value being -0.30 this will result in the optimizer stopping at around -0.15. While the peak for the window size of 7 pixels is closer to the ground truth value, the graph is flat for a large range of values and a small error could set the peak at -0.5.

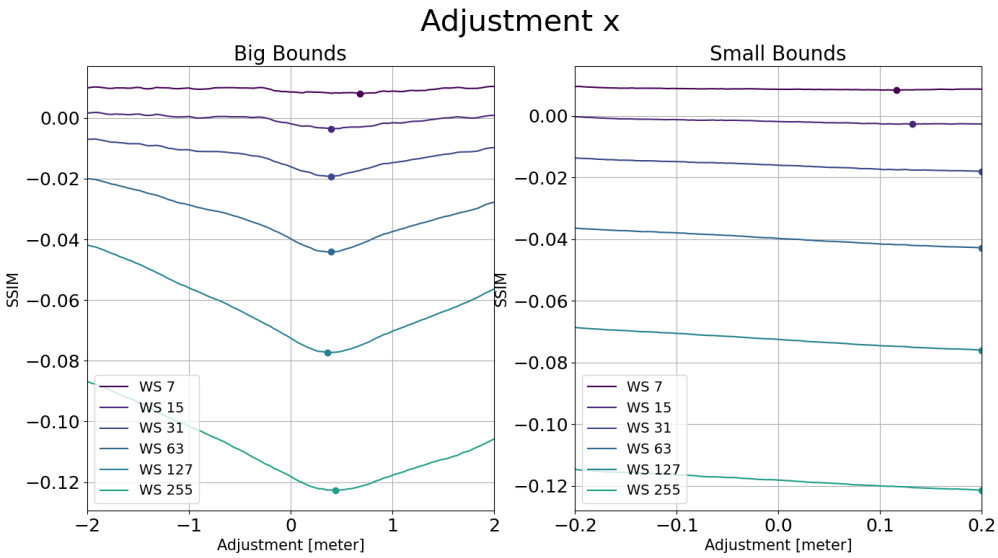

Figure 11: SSIM Evaluation X-axis

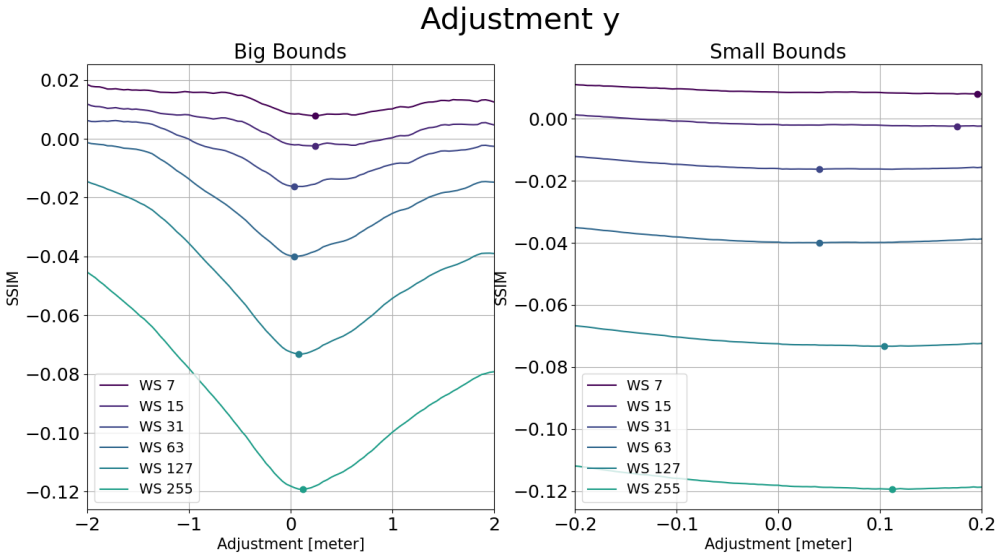

Figure 12: SSIM Evaluation Y-axis

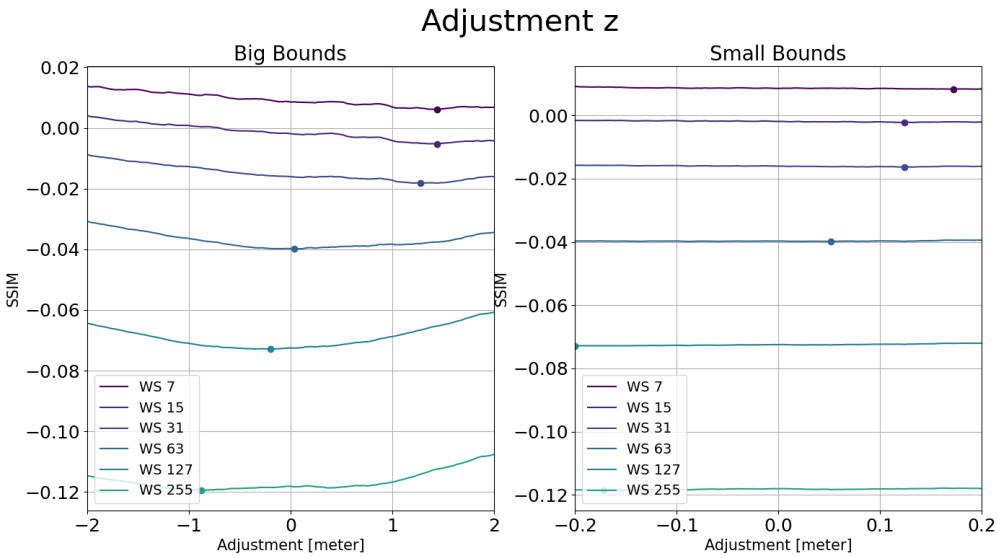

Figure 13: SSIM Evaluation Z-axis

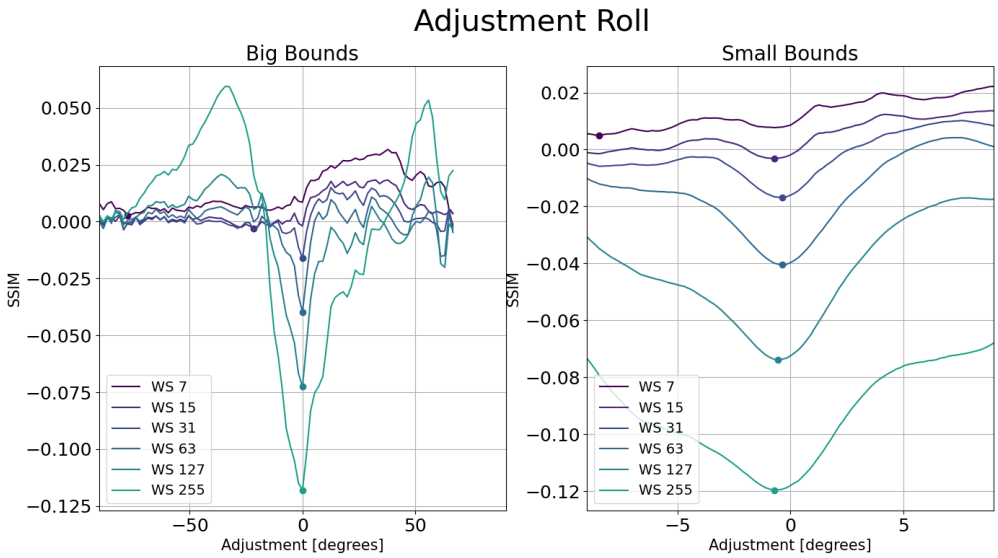

Figure 14: SSIM Evaluation Roll

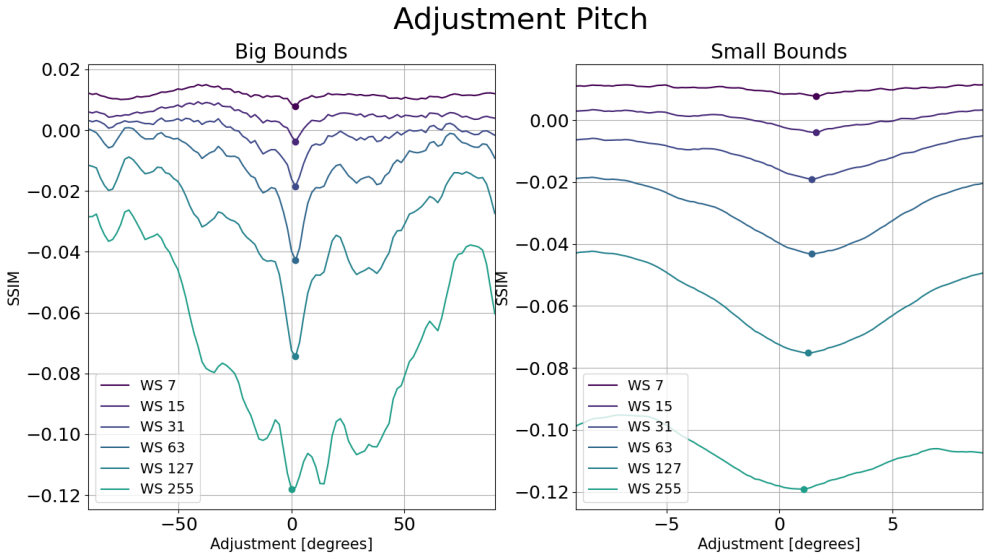

Figure 15: SSIM Evaluation Pitch

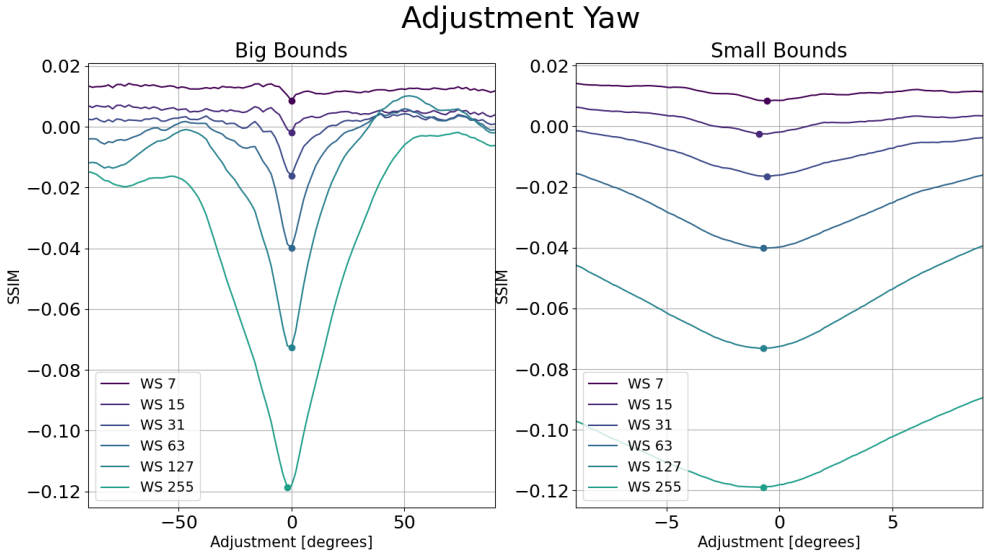

Figure 16: SSIM Evaluation Yaw

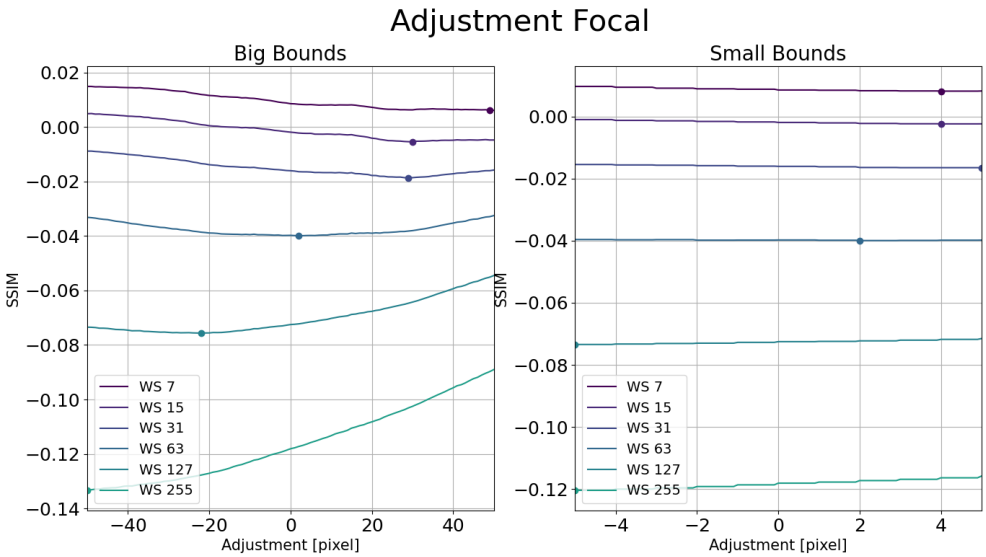

Figure 17: SSIM Evaluation Focal length

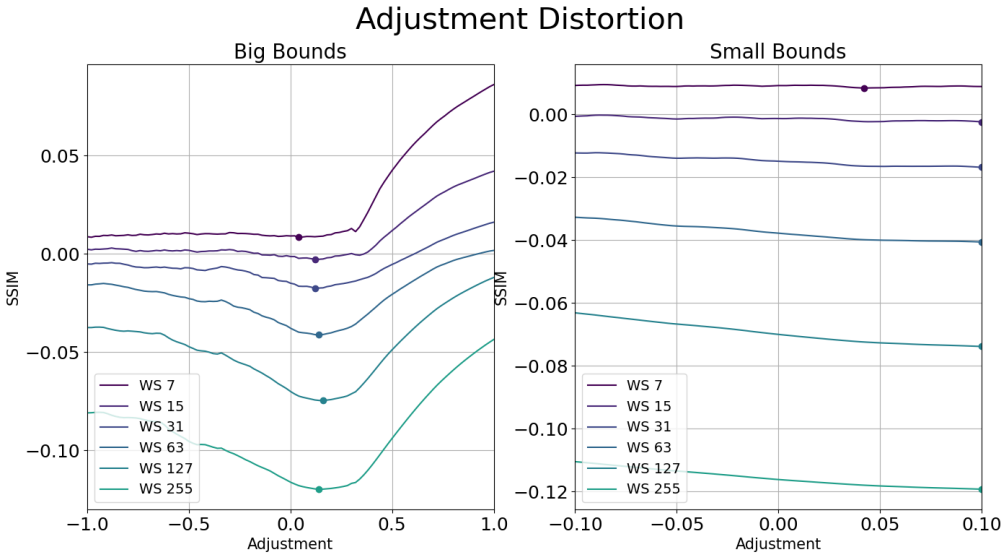

Figure 18: SSIM Evaluation Distortion

From these SSIM plots, it can be seen that there are peaks for almost each window size and axis, these peak values are used to adjust the ground truth values with which the depth images are made, and the camera images are rectified. All of these tests are also done to the NTU sbs_01 Kitti and handheld datasets, the plots for these can be seen in appendix B.
To see the effect of the environment on the SSIM evaluation, the evaluation is also done on the NTU sbs_01 dataset, since it has the same sensors as the NTU eee_01 dataset, all the differences come from the environment. In figs. 28 to 30, the SSIM evaluation plots for the translation can be seen. The graphs look similar to those of the eee_01 dataset, only the peak values are not as close to the ground truth value, especially for the Y-acis the points are more off, and for the smaller window sizes. In figs. 31 to 33, the SSIM evaluation for the rotational axis can be seen. These plots look different than the eee_01 plots for the same axis, especially for the Roll and Pitch, here the peaks are less defined. The Roll axis has more noise and for the biggest window size of 255, the peak is at an error of 50 degrees. Also around the adjustment of zero for the Roll axis, there are 2 peaks with at an adjustment of zero the SSIM value is higher which is strange. The evaluation for the Yaw-axis is the most similar to the NTU eee_01 evaluation. In figs. 34 and 35 the Focal length and Distortion SSIM evaluation is shown, here also can be seen that the Focal length there are no visible peaks. The Distortion evaluation has now also less defined peaks and for the bigger window sizes between an adjustment of 0 and positive 0.3 there is a flat spot/peak. In table 9 the error between the peak value and the ground truth is shown, here can be seen that some of the errors for the window of sizes 7, 15, and 255 pixels are not good enough to estimate the rotation and for the smaller two not even for the translation.
The translation SSIM evaluation for the Kitti dataset can be seen in figs. 36 to 38, in these plots, there is not peak visible even for the left plots with bigger bounds. For the Y-axis there are some small peaks, but the biggest window size has its peak at 1.25 meters difference with the ground truth value. This makes it impossible to do the translation calibration for the Kitti dataset. For the rotational SSIM evaluation, which can be seen in figs. 39 to 41, the peaks are visible for bigger bounds, but these are less defined in when zooming in on the 0 adjustment. The Yaw rotation has the best defined peaks. For the Intrinsics parameters SSIM evaluation, which can be seen in figs. 42 and 43, here all the graphs are flat for both the Focal length and Distortion. The peak value for the Distortion the peaks are at 0 for the bigger window size, and zooming in per graph a small peak is visible but this peak is small. This means the SSIM change per value change is small, so the optimizer for the distortion should have a low-value termination value. So the calibration will not be able to estimate the translation and the Focal length, the Rotation, and the Distortion parameters will be difficult but may be possible.
In figs. 45 to 47 the SSIM evaluation for the translational parameters are seen, and in figs. 48 to 50 for the Rotnational parameters. These are more comparable to the NTU dataset, with some peaks visible for bigger bounds, but when zooming in onto an adjustment of 0 the peaks flatten out for the translation but for the rotation still visible. For the Intrinsics SSIM evaluation, in figs. 51 and 52, the graphs look the same as in the Kitti dataset with almost all graphs flat. This results in the optimizer not being able to determine the translation and focal length, the distortion will be difficult but the rotation may be possible.
This SSIM evaluation can be considered a calibration process for a single degree of freedom, where the optimizer seeks the lowest value, corresponding to the peak. Table 1 shows the error (calculated as the absolute difference between the peak and the ground truth value) for each of the eight parameters across different window sizes, with the lowest error in each column highlighted. From this table, it is evident that a window size of 63 results in the lowest errors for most parameters, and even for those where it is not the lowest, the difference is minimal. Similar error evaluations were conducted for the Kitti and Handheld

combination datasets, as shown in Tables 10 and 11. For the Kitti dataset, the optimal window size varies, with both large and small window sizes yielding the lowest errors for different parameters. In contrast, for the Handheld dataset, larger window sizes consistently perform better.

Table 1: Error value for one degree of freedom SSIM-calibration for the NTU eee_01 dataset

Window Size [pixel]	x [m]	y [m]	z [m]	Roll [°]	Pitch [°]	Yaw [°]	Focal [pixel]	Distortion
7	0.68	0.24	1.44	77.40	1.62	**0.54**	49.00	**0.04**
15	0.40	0.24	1.44	0.72	1.62	0.90	30.00	0.12
31	0.40	**0.04**	1.28	**0.36**	1.44	**0.54**	29.00	0.12
63	0.40	**0.04**	**0.05**	**0.36**	1.44	0.72	**2.00**	0.14
127	**0.36**	0.10	0.20	0.54	1.26	0.72	22.00	0.16
255	0.44	0.11	0.88	0.72	**1.08**	0.72	50.00	0.14

In the table 1 the adjustment values of the peaks are shown, with the adjustments for a window size of 63 a depth and camera image are made which can be seen in figure 19. The left depth image is made with the ground truth and the right depth image is made with the optimal values, there is a difference that the optimal depth image is made a bit to the right but no further difference can be made. There are three camera images, the raw camera images, the camera image rectified with the ground truth values, and the rectified with the optimal values. In the raw camera images, there is distortion, which can be seen that the building is curved which should not be. In the peak value camera image, there is still some curvature to the building and some parts of the towers on the side are visible, which is not there in the ground truth camera images and the towers are not visible.

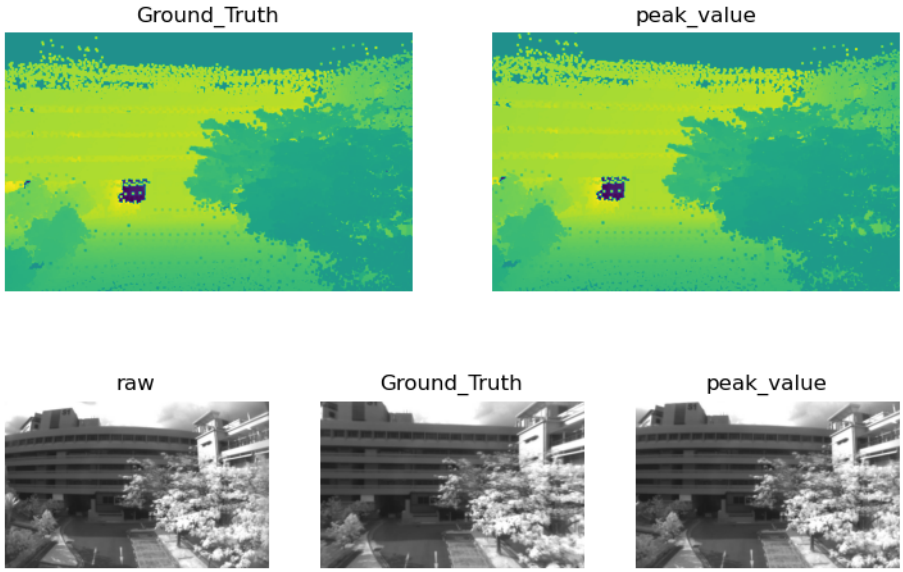

Figure 19: NTU eee01 ground truth and peak depth and camera images, for a window size of 63

4.3 Calibration Results

To see if it is feasible to do the combined extrinsic and intrinsic calibration with the use of SSIM, first, the extrinsic calibration is done separately. The calibration for the individual parameters is done mostly with the NTU eee_01 dataset, but later on, the other datasets are also used to compare the results and see the generalization of the approach to different environments.

4.3.1 Extrinsic Calibration

The extrinsic calibration is performed by varying the initial transform within specific bounds. When the translation is adjusted, all three axes are modified simultaneously, and the same applies to the rotation. This ensures that changes occur in 3 degrees of freedom (3DOF, 6DOF) rather than any other number of DOFs. The augmentation bounds are set to a maximum deviation of 20 cm for each of the translation axes and 10 degrees for each axis of rotation. Table 2 presents the average errors in translation (Euclidean distance, in meters) and rotation (quaternion distance, in degrees), along with their corresponding standard deviations. Additionally, the mean error for individual axes and the time required per image pair are added. The table includes four rows, corresponding to four optimization types: translation and rotation (6-DOF) or only rotation(3-DOF), each calibration is done with either distorted or rectified images. When only rotation is optimized, translation remains unchanged and it is at the ground truth value and is therefore omitted. The distorted images refer to the raw camera images from the dataset, while the rectified images in the first two calibration types are undistorted using ground truth intrinsic parameters and distortion coefficients.
The data in the table 2 indicates that the error in translation remains Big. After examining the initial and optimized values, it was noted that the translation parameters are not optimized and remain close to their initial values. This is further corroborated by the error observed for each axis, which consistently hovers around 10 cm. When adjustment values are randomly sampled within specified bounds using a uniform distribution, the mean absolute error is expected to be approximately half the bound. For instance, with bounds set at $\pm 20cm$, the resulting initial error is approximately 10 cm per axis, leading to an Euclidean distance error of around 17 cm. Given this context, it is clear that the translation values are not being optimized during the calibration process.
The Rotation does get optimized more than the translation, and the optimized value is for most tests not around the initial value. The error is better, but it is still not good. From the error per axis, it can be seen that the pitch has always the biggest error, while the Roll has the lowest. These calibrations also take some time to do, with the lowest being 54 seconds per image pair, with 50 images this takes a lot of time to do. With more degrees of freedom, so more search space, the time it takes to find the optimal values is also higher. Similar results are seen for the sbs_01 dataset, for which the results can seen in table 3. The translation is still not optimized and has similar values as for the eee_01 dataset, the rotation is here slightly higher by around 1 degree per test.

Table 2: Extrinsic calibration results, max augmentation ±20cm and ±10° for the NTU `eee_01` dataset. Euclidean distance for translation and Quaternion distance for rotation, and also the error per axis for the calibration.

Calibration Type	Translation error [m]				Rotation error [°]				Time [s]
		X	Y	Z		Roll	Pitch	Yaw	
Translation & Rotation Rectified	0.21±0.05	0.11	0.08	0.12	7.0±3.4	2.8	4.1	3.4	111
Rotation Only Rectified	-	-	-	-	6.6±3.6	2.8	4.2	3.2	65
Translation & Rotation Distorted	0.17±0.05	0.09	0.07	0.10	7.6±3.9	3.1	4.0	3.8	76
Rotation Only Distorted	-	-	-	-	7.5±4.3	2.8	4.4	4.0	54

Table 3: Extrinsic calibration results, max augmentation ±20cm and ±10° for the NTU `sbs_01` dataset. Euclidean distance for translation and Quaternion distance for rotation, and also the error per axis for the calibration.

Calibration Type	Translation error [m]				Rotation error [°]				Time [s]
		X	Y	Z		Roll	Pitch	Yaw	
Translation & Rotation Rectified	0.20±0.04	0.13	0.08	0.10	8.0±2.4	3.4	5.4	3.9	85
Rotation Only Rectified	-	-	-	-	7.6±2.7	3.6	3.9	4.1	51
Translation & Rotation Distorted	0.18±0.07	0.13	0.07	0.08	8.9±2.4	3.3	4.6	6.0	85
Rotation Only Distorted	-	-	-	-	8.6±3.3	3.2	4.1	5.5	46

4.3.2 Intrinsic Calibration

From the SSIM evaluation it could be seen that for the focal length, at least on its own, it would be impossible to do the calibration. For this reason, only the distortion calibration is tested, and later a combination of focal length and distortion to see if the combination has a possible effect. In table 4 the intrinsic calibration results can be seen, At least 10 different tests are done, and the mean and standard deviation from these tests. The distortion is calibrated for, with the focal length set to the ground truth values, only the K1-distortion parameter is optimized for, the initial value is now set at 0. The results can be seen in table 4, and the initial and resulting distortion values can be seen in figure 20. The distortion has an average error of 0.2 with a small variance and it takes an average of 18 seconds per image pair to do the calibration which is the lowest of all the calibration types. In the figure, the resulting distortion values all go the the ground truth value but do not reach the value and stop halfway. Then 10 tests are done where both the Focal length is augmented and the K1-distortion value is set to zero. The average results can be seen in the last row of table 4. The error in the distortion is smaller but took a bit longer to optimize, the focal length error is not added since it just depended on the initial value and is not optimized as expected. The Initial and resulting focal length and distortion value for each test can be seen in figure 21, the resulting focal length does change a little more compared to when only the focal length is optimized but all go down and do not seem to converge to the ground truth or any other value.

Table 4: Intrinsic calibration results

Calibration Type	Augmentation [Focal length,K1]	Distortion	mean Time per image pair [s]
Distortion Only	[-, K1=0]	0.2 ± 0.055	18
Focal & Distortion	[$\pm$10 K1=0	0.190 ± 0.027	38

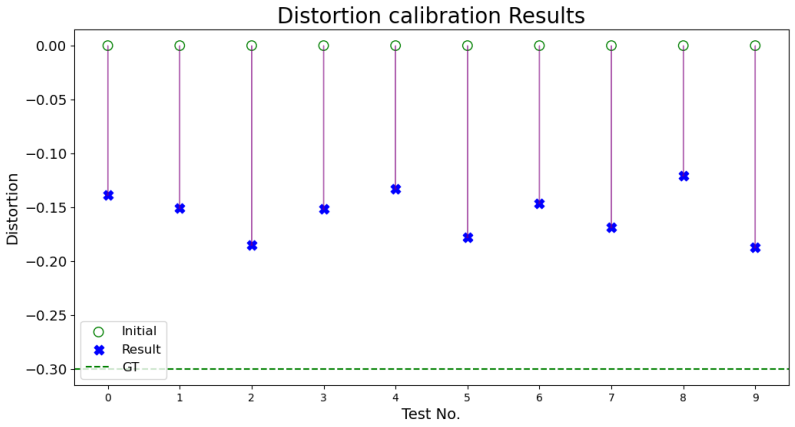

Figure 20: plot of the initial(Cirlce) and resulting(Star) Distortion for different augmentation bounds

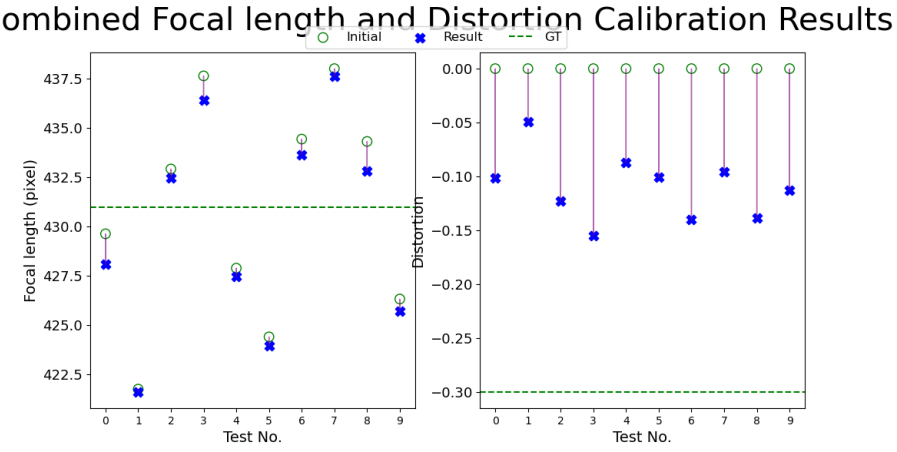

Figure 21: plot of the initial(Cirlce) and resulting(Star) Focal length and Distortion, (Keep in mind similar resulting distortion could have different resulting focal length)

4.3.3 Combined Extrinsic and Intrinsic Calibration

The extrinsic and intrinsic calibration are combined, with the extrinsic calibration on distorted images first and with the wrong initial focal length. After the extrinsic calibration is done the intrinsic calibration is done with the optimized transform. From the previous results, it was seen that translation values remained largely unchanged during optimization, and the estimated focal length has the same issue. Because of this also the combined calibration is only done for the rotation and distortion, with GT translation and GT focal length. These calibrations are done on the four different datasets. The result is the error in the transform between the imu and the camera and the intrinsic, to get the error between the LiDAR and Camera, the error between the LiDAR and IMU will need to be added. But since to know how well this calibration works, and not be affected by error from other algorithms this is not done.
The results of the calibrations can be seen in table 5 for all the Extrinsic and Intrinsic parameters. It can be seen that the translation and the focal length do not get optimized much, since the error is still as big as in their calibration, and does not differ much between datasets. The Rotation is better optimized for the NTU datasets but this is around 3 degrees difference, but when only the rotation is optimized the values are closer but still the NTU has a lower average error. The distortion is better estimated when also the focal length is optimized, especially for the Kitti dataset. The time it takes to do the calibration is longer when all parameters are optimized than when only the rotation and distortion are optimized, is around 3 times longer.
In table 6 the results for only the rotation and distortion are shown, the time it takes to calibrate only these 4 parameters (3 rotation and distortion) is a lot less even less than half the time. The Rotation results are better but still not good and almost equal for all datasets but the distortion is worse, especially for the Kitti dataset.

Table 5: Extrinsic & Intrinsic calibration results, euclidean distance for the translation, Quaternion distance for the Rotation error, and absolute difference for the Focal length and distortion.

Dataset	Translation error [meter]	Rotation error [°]	Focal Length error [pixel]	Distortion error	mean Time per image pair [s]
NTU eee_01	0.19 ± 0.05	9.1 ± 4.4	6.1 ± 3.3	0.262 ± 0.055	100
NTU sbs_01	0.19 ± 0.05	8.5 ± 2.5	5.0 ± 3.2	0.219 ± 0.078	87
Handheld	0.21 ± 0.08	12.0 ± 3.0	4.3 ± 3.0	0.146 ± 0.102	162
Kitti	0.20 ± 0.05	11.2 ± 3.3	5.2 ± 3.5	0.346 ± 0.060	123

Table 6: Rotation & Distortion calibration results, Quaternion distance for the Rotation error and absolute difference for the distortion.

Dataset	Rotation error [°]	Distortion error	mean Time per image pair [s]
NTU eee_01	8.3 ± 2.9	0.462 ± 0.379	53
NTU sbs_01	8.3 ± 4.4	0.223 ± 0.088	33
Handheld	9.6 ± 3.0	0.328 ± 0.451	50
Kitti	8.6 ± 1.6	1.690 ± 0.787	47

4.4 Benchmark

In table 7, the results of other approaches, these approaches use the Kitti dataset to train their models. In the table, our results of the kitti dataset calibrations are added. It can be seen that their results are much better and get better accuracy for the rotation and distortion for the CaLiCaNet, and all also estimate the translation, which our approach does not do at all.

Table 7: Benchmark result for our model compared to other approaches

	Translation loss[m]	Rotation loss [°]	Focal Length Loss [pixel]	Distortion Loss	code open source
Ours	-	8.6	-	1.690	Yes
CaLiCaNet	0.059	0.154	11.02	0.067	No
Calibnet	0.0434	0.41	-	-	Yes
CalibRCNN	0.093	0.805	-	-	Yes
joint-lidar-camera-calib Li et al. (2023)	00.0383	0.13	-	-	Yes

4.5 Calibration time

To determine how many image pairs are required for calibration, it is needed to evaluate the calibration performance as a function of the number of image pairs and the time taken for the calibration process. Table 8 shows the results. The average rotation loss decreases as the number of image pairs increases. The mean time per image pair remains relatively stable, except when 100 image pairs are used, where the time per image pair is significantly lower, resulting in a shorter overall calibration time compared to larger sets. For all the previous tests, 50 images are used, because it gives the best results together with 100 images, but is less resource intensive.

Table 8: Calibration result and time for a different number of image pair used

Num Images per Test	Num Test	Rotation error [°]	Mean Time per image pair [s]	Mean Time per Test [s]
1	13	9.5 ± 3.3	47.8	48
5	13	9.7 ± 4.0	61.7	309
10	15	9.6 ± 3.4	58.2	582
25	16	8.7 ± 4.9	41.0	1024
50	33	7.1 ± 4.0	59.0	2952
100	16	7.1 ± 5.4	21.6	2159

5 Discussion

This section begins by analyzing the results presented in the Results section, following the same order. Both the positive and the negative outcomes are addressed, along with potential reasons for any observed errors. Following this, broader aspects of the project are examined, including error analysis, challenges encountered, limitations of the approach, and potential areas for improvement. Finally, ideas for future work and extensions of the methodology are explored.

5.1 Results Discussion

5.1.1 Checkerboard calibration

The ground truth for the Handheld needed to be determined, this was done by recording data from a checkerboard that is on a computer screen. This was done because initially there was no physical checkerboard nearby and since a monitor is flat it could be a good way to determine the intrinsics since screens are everywhere.
The intrinsic calibration resulted in a low reprojection error, and comparing it to the manufacturer values, the result seems to be good, and no issue there. It has an error that not the correct size checkerboard can be found in any of the images, but only a pattern with a border of two squares around it. While it is not known why this error happens, it is most likely the issue of the black border around the checkerboard from the screen and not a white border.
The Automatic approach for the extrinsic calibration is done by matching the camera points with points in the LiDAR data, these are matched by finding the checkerboard corners in the camera image. Then the pointcloud data is processed by first finding the plane, determining the sides of the plane, getting the intersections of the side to get the corners, adjusting for the housing to get the 4 corners of the checkerboard, and then getting the same corner pattern as in the camera image. This is a 20 by 9 grid, the same grid as the checkerboard corners in the image. With these matching points and the SolvePNP function, the extrinsic calibration was determined. From the results, it could be seen that the rotation is almost the same as the transform determined by hand with right angles between the sensors. The translation is not well calibrated with this method, the distances are not realistic except for the Y translation of 8cm. The reason for this is that the corners of the plane found in the lidar data are not good enough and this results in all other points being off. The reason the sides are not detected well is because sometimes a corner is missing or a scan lines line up so that it misses the edge of the screen. This was tried to reduce to first expand the corners to the size of the screen, then remove the padding of the housing but this did not help. Also for good calibration, the points should be matched in as many poses as possible, but when the sensor setup was rotated around the camera's Z-axis then or the checkerboard is not detected in the image, or the plane is not fully recorded. When one or the other happens the data is rejected and in total there were only one or two data samples with a rotation around the camera's Z-axis.
Since the approach in Python did not work, an existing method in Matlab was used. Here the resulting transform is much better and is much more comparable to the initial derived transform, but there is still a small error in the translation. For the Hesai Pandar XT-32, the LiDAR sensor is $46mm$ above the base, with this translation it does not even get out of the LIDAR. The reason for this is that when inspecting the reprojected points, several points are outside the screen, but this can also be because of wrong point selection.
Checkerboard calibration approaches normally work well, however, using a checkerboard displayed on a screen instead of a physical plane introduced several challenges, which were not expected. When viewed at an angle, the screen's edges become visible, making the

detected plane larger. This caused errors in the data, as it became unclear where the screen's sides ended and its front began. Additionally, a speaker located below the screen occasionally interfered with size detection, being sometimes included in the found plane and other times not. This would then increase the size and the found points would not be the correct corners that match with the chekerboard grid. Also, other approaches use the difference in intensity of returned points when it hits a black or a white square, but with the screen, there is no visible difference and only some artifacts from the lights in the room. Furthermore, the screen's partial transparency sometimes allowed points to go through the first layer and reflect from a layer behind it rather than from the screen itself. These factors made it difficult to accurately determine the correct 3D points required for matching with the camera image and resulted in the bad checkerboard calibration.

5.1.2 SSIM Evaluation

The SSIM evaluation for one degree of freedom at a time for the translational adjustments can be seen in figs. 11 to 13. In the left plot with the large bounds of $\pm 2meter$, there is a minimum, or at least for the higher window sizes of 63 and above, but these peaks flatten out when the bounds get smaller. The peaks, highlighted with the dot, are for the adjustment along the X-axis not near the adjustment of zero or the ground truth but more around an adjustment of plus half a meter. The error for the Z-axis is for small window sizes more then a meter in the positive direction, then at a window size of 63 almost at zero, and then for bigger window sizes towards an error of a meter in the negative direction. This is strange and looking at the image no good explanation was found. With table 1, it can also be seen that the window size of 63 is the best window size for most of the parameters, a window size of 31 is also possible since the difference is minimal between the two sizes. This was for the NTU dataset, but for the Kitti smaller window size would be better to use, which can be seen from table 10. This is most likely because the depth environment is bigger and a small window will still contain a lot of data. For the Handheld combination, it is the other way around, the larger window size is better, this is because the depth environment is smaller compared to the other datasets, but still here 31-63 is a good window size, since too big may take too much of a mean value that all the small details are reduced.
The evaluation was done multiple times, and every time similar results were retrieved. The peaks not being at the ground truth may be because of possible errors in the Odometry, possible errors in the pointcloud and so in the depth image, or too similar environment that small adjustments together with other errors set the peak not at an adjustment of zero. The SSIM value also does not change much for small translational adjustments, which can be seen in the right plots, this is because the depth image will change a really small bit, especially when the images are taken in an open environment with objects far away. Then the image will still look the same and the optimizer will not be able to optimize the translation. So to possibly better get the translation in an environment with objects closer to the camera and the LiDAR is needed.
The peaks in the SSIM evaluation for the rotational components of the transform are more defined, these plots can be seen in figs. 14 to 16. In the evaluation, peaks are visible, even for the small bounds, especially for bigger window sizes. These peaks are also at or near an adjustment of 0. The reason the peaks are now more defined than in the translational SSIM evaluation is that a small adjustment in the rotation gives a bigger difference in the field of view. The peaks for the roll and pitch are not perfect at 0 adjustments, this may be of the same errors as with the translational evaluation or slightly wrong ground truth rotation. Since the peaks for the rotation are more defined and closer to an adjustment of zero than for the translation, the calibration will also be tested with only the rotational components.
The SSIM evaluation plot shown in figure 17 should reveal a pattern similar to that of the

translational adjustment along the y-axis, since, as the focal length determines the distance between the image plane and the camera sensor, impacting the apparent scale and alignment of the image features. Similarly, translation along the y-axis changes the effective positioning relative to the image plane, comparably influencing the creation of the depth image. Therefore, adjustments to focal length and y-axis translation both have overlapping effects. But this is not the case, when looking at the plot it seems that for the bigger window sizes the peak is well below the ground truth, for the smaller window sizes there is no defined peak. The best window size for the focal length is also 63, but there is no good gradient here, and will be impossible to optimize the focal length this way. For this reason, the focal length was not optimized for on its own.
The Distortion SSIM evaluation, in figure 18, has similar aspects as the other plots with the biggest window size having the lowest SSIM value. The peaks are not at an adjustment of zero, the graph of the window size of 63 has its lowest peak close to zero adjustment while still having somewhat of a peak, but the graph is not as smooth and the peak is not much different from the other values on the graph. This evaluation is only done with the NTU dataset images, the K1-Distortion coefficient of the camera is around -0.3, so the images have barrel distortion. To make the evaluation better, more images from different distortions and distortion types should be used. Since the scope only uses the K1-distortion coefficient, and not more the image is not fully undistorted, which can be seen in the corners of figure 22. Under the subplot title is the SSIM value of that image with the depth image. The values of undistorted images are almost the same but both a much different than the raw image ssim value. The optimization will not fully find the undistorted image, but if the optimization finds the correct value for K1 then the undistorted image will be better than the raw image.

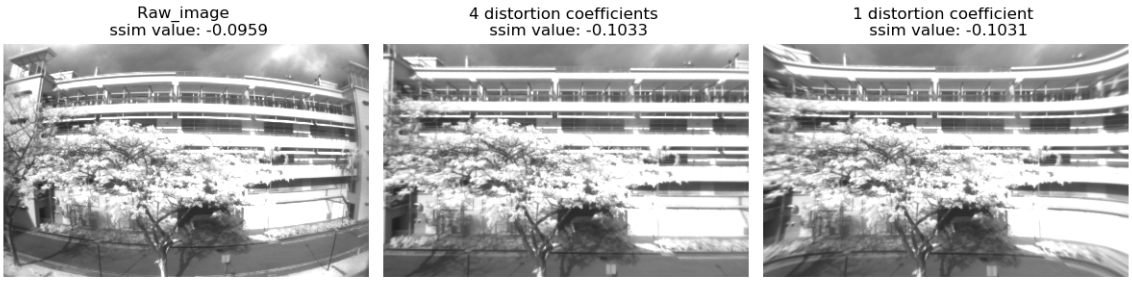

Figure 22: Image undistortion with all (5) or 1 distortion coefficients

All of these evaluations were also done for the NTU sbs_01, Kitti, and handheld datasets, which can be seen in section B in the appendix. All of these are made in different environments and with different sensor systems, except for the NTU datasets which are made with the same dataset but in different environments. For the evaluation along the translational axis, there are some slight changes between the datasets but for all the main result is that some peaks are visible when looking over a large range of adjustments, but when zoomed in to around the adjustment of zero the gradient of the graph is almost zero. So for all the datasets, it is impossible to accurately calibrate for the translation with this approach, it will still be calibrated to see the effect with both a translation and rotation of the calibration. For the Rotation SSIM evaluation, the graphs for the Handheld evaluation give similar results as those for the eee_01 dataset, here the peaks are at or close to the ground truth values and the peaks are still (slightly) visible for the narrow bounds. For the sbs and the Kitti, the results of the evaluation are worse, here there is more noise in the graphs, the

peak values have a bigger error or the peaks are less defined even for the bigger bounds. The focal length has the same results for all the datasets, it does not have a peak and it is impossible to do the calibration for the focal length on its own. The Distortion evaluation also looks the same for the different datasets, with a high SSIM value for the big positive adjustments, for smaller adjustments or negative adjustments the graph is almost flat for the Kitti and the handheld datasets, for the NTU dataset there is a small dip but the peak value has some error in it. From all of these results, it can be concluded that the calibration will be impossible for the translation and the focal length parameters, but it may be possible to do the calibration for the rotation and the distortion. Per the dataset is that the calibration will be the best for the NTU eee_01 dataset since it has the lowest error for the individual DOF for the rotation and also the distortion (with a window size of 63), then the error for the NTU sbs_01 dataset will be higher, and for the distortion, a bigger window size should be used. For the Kitti dataset, a smaller window size of 31 should be used for the rotation, but for the distortion, a bigger window size of 31, the reason is that the Kitti dataset depth environment is bigger and a small window size will have enough information but not too much, the distortion window size should be bigger since the distortion is all of the image and not just a small part of the image. For the Handheld dataset, a window size of 63 should be used for the rotation and distortion, while a window size of 127 is also not bad for the rotation. These are calibrations of a single degree of freedom at a time, but calibration is done for more DOFs.

5.1.3 Extrinsic calibration

The extrinsic calibration was conducted using the NTU datasets, with two approaches: optimizing both translation and rotation and optimizing only the rotation. Each approach was done with both distorted (raw) and rectified camera images. The results can be seen in Table 2 for the NTU eee_01 dataset and Table 3 for the NTU sbs_01 dataset. The results show that translation estimation yielded an average error between 17 and 21 centimeters for the Euclidean distance, and around 10cm error per axis. These values are the average initial error since the transform is augmented with $\pm 20cm$ with a uniform distribution, this results in an average error of 10cm per axis, and with 3 axes the Euclidean distance of around 17cm. All of this is to say that the translation does not get optimized. The rotation does get optimized better, the average initial error would be around 5 degrees for each axis and 8.5 degrees for the combined axis, but the resulting error is between 2.8 and 4.4 for the NTU eee_01 dataset. This indicates that the rotation does get optimized. From the table, it can be seen that the pitch is optimized the least, for both the NTU eee_01, the reason for this is that the pitch seems to have the lowest gradient of the three rotations. It was attempted to visualize the error and see if the optimizer was stuck in a local minima, but with the multiple DOFs that are changed, it was not able to be done in a way good way. For the NTU sbs_01 dataset, the error in the rotation is bigger, especially for the Yaw axis with distorted images, the mean error is even bigger than the average initial error. This indicates that the optimizer has more difficulty doing the calibration for the sbs_01. Overall the optimizer can not optimize the translation, can optimize the rotation a bit better but the resulting error is still too big. A possible reason for this is that the gradient in the SSIM graph is much smaller for the translational components compared to the Rotational components, which makes the optimizer focus on optimizing the rotation more than the translation.
The calibration process takes a lot of time to do, with many runs exceeding 20 minutes. This is largely due to the computational demands of repeatedly generating depth images with updated transforms during optimization. Depth image generation is performed using Open3D, which does not have full GPU support for the required functions, resulting in slower performance. Accelerating this process would likely require an alternative method

for generating depth images from point clouds, but time constraints prevented exploration of this avenue. Despite the lengthy process, the approach does achieve relatively better rotation estimation compared to translation.

5.1.4 Intrinsic calibration

The intrinsic calibration was done using the NTU eee_01 dataset, calibrating the distortion parameter, and later a combination of the focal length and distortion. The focal length was not done separately because in the figure 17 of the SSIM evaluation for the focal length it could be seen that it would be impossible to get the correct value for the focal length. Since to undistort the image uses the focal length, maybe with a combined calibration with the distortion, it could be possible to do the calibration.
For distortion, Figure 20 shows that all initial K1 distortion values, which were initialized at zero, optimized towards the ground truth but all stopped halfway. This comes from that the peak value of the distortion is also not at the ground truth value but at a more positive value. So the optimizer found the minimum value, but this is the wrong value.
In Table 4 the long time required for intrinsic calibration when both focal length and distortion were calibrated together can also be seen. While the combined calibration resulted in slightly lower errors and reduced variance for both focal length and distortion, these improvements are small and likely because of lower initial errors rather than better optimization.
In Figure 21, the initial and resulting values for both focal length and distortion are visualized. The focal length and distortion are optimized at the same time, but due to differences in their required step sizes and termination thresholds (Ftol) a problem was found. For the focal length, a step size of 1 to 5 was found to be suitable, but this step size was too large for the distortion. Similarly, the Ftol value for focal length optimization was set to 10^{-4}, while distortion required a much smaller value of 10^{-25}, determined through trial and error. To balance these differences during combined optimization, a scale factor of 20 was applied to the distortion updates, which yielded reasonable results. The focal length changes slightly more when combined with distortion calibration, but the values go downward instead of converging toward the ground truth. Even with combined with the distortion the focal length can not be optimized, and another way should be determined to get a good estimation of the focal length.

5.1.5 Extrinsic and Intrinsic calibration

For the combined Extrinsic and intrinsic calibration two types were done, one with all the parameters to see how the calibration is affected and then also done by optimizing the rotation and the distortion only. From individual extrinsic and intrinsic calibration was seen that the translation and focal length could not be optimized which was also seen in the SSIM evaluation plots, the gradient was low or did not exist. This was also the result of the combined optimization.
The results of the calibration where all the parameters are optimized can be seen in table 5, the translation and the focal length do not get optimized, as was expected from their SSIM evaluations. The error in the rotation is now bigger than when only the rotation is optimized, it looks like for most of the dataset the optimizer makes the error bigger. The mean initial value would be on average around 8.5 degrees but these are much higher. The reason for this can be that the translation and the wrong focal length have a big influence on the rotation optimization. Or that the combination of all of these parameters changes the gradients of the graphs for the individual axis so much that it adds new local peaks. The error in the Distortion is now slightly higher, but the extrinsic calibration is not good

and this will result in the distortion also being less good. These optimizations were time-consuming. On average, the calibration process with 50 image pairs took approximately 72 minutes for the SBS dataset and over 2 hours for the Handheld dataset. This duration was impractical, limiting the number of tests that could be conducted. Table 6 shows the calibration results when only rotation and distortion parameters are optimized sequentially. The rotation values across different datasets are relatively similar, and the distortion values are similar except for the KITTI dataset, which has significantly higher distortion and variance. This increased variance indicates that the results are not skewed by a single outlier but are more broadly distributed.
Based on the SSIM evaluation for each dataset, it was anticipated that the NTU me and Handheld datasets would have the lowest rotation error, followed by the sbs dataset with a slightly higher error, and the KITTI dataset with the highest error. However, the results do not show these expectations. Surprisingly, the Handheld dataset shows the largest rotation error, despite its SSIM evaluation revealing distinct peaks. This discrepancy is likely due to the optimizer struggling to locate the minimum SSIM value when combining different rotations, making the optimization process more challenging.
The average rotation error for the two NTU datasets is identical, but the SBS dataset has a higher standard deviation. This could be because of the less defined peaks in the SSIM evaluation, which make precise optimization more difficult. Although the KITTI dataset has the largest overall rotation errors in the SSIM evaluation, its error decreases when all three rotations are optimized simultaneously. This could be a coincidence, where the combined optimization brings the peak closer to the ground truth. However, the KITTI dataset also exhibits the highest distortion error—over one unit higher than other datasets. This could result from rotation errors influencing the intrinsic calibration.
In conclusion, while this optimization approach shows some potential, it currently has significant challenges. The calibration results are inconsistent, and the method does not yet produce reliable outcomes.

5.1.6 Benchmark

This approach is compared to other state-of-the-art methods and this can be seen in table 7, the result for when only rotation distortion is calibrated for is compared to the other approaches. Here it is assumed there is no error in the LiDAR to IMU calibration, this is because wanted to compare the result of this calibration and this approach is not dependent on the Lidar IMU init since it can be done by other methods which may work better for certain scenarios. It can be seen that our approaches are really bad compared to all other approaches shown in the table, the translation and the focal length are not stated in this table because they do not get optimized and only depend on how good the initial guess would be.
Overall this approach is not good, and can not calibrate to the accuracy that is needed to be used for other applications. This approach may be a better use to get a general initial calibration, with which other applications can be used to get the finner calibration. This is if something is not changed with this approach to make it more accurate, like a better optimization or a different comparison function than SSIM.

5.1.7 calibration time

The Calibration error and the time it took were tested, to see which number of images was the best to use in terms of time, accuracy, and resource use. For this test, the Rotation calibration was chosen because this gave the best calibration results. From table 8, it can be seen that the accuracy gets better with the number of images used, but also the overall mean time it takes to do increases. But the time per image does not get lower or much

higher than with more images, except for 100 images, and there is no clear trend in the timings. The reason for this is that it just takes a certain amount of time to make the depth image and then compare it, with some overhead time added. The reason for the lower time for when 100 images may be the low sample size, together with possibly running on a better server with better hardware. Most of the tests are run on all of the servers so this should remove the effect of this on the calibration time or reduce it. But to not use too many resources, 50 images are used for the other tests because it gave a lower variance, the same error, and uses fewer resources.

5.2 Error Analysis

To assess the accuracy and reliability of the calibration results, a good error analysis is needed. Given the complexity of intrinsic and extrinsic calibration processes, multiple error sources must be examined throughout the calibration pipeline, beginning with data measurement.

- **LiDAR Measurement Noise**: LiDAR measurements are influenced by environmental conditions (for example, rain, fog, or dust) and hardware limitations, such as range and resolution. Additionally, LiDAR beams may pass through transparent surfaces (for example, windows) or reflect off mirrors, creating wrong points in the point cloud. Filtering can reduce but not eliminate these residual errors. The Hesai Pandar XT32 LiDAR reports an accuracy of ± 2 cm Hesai Technology (2021), assumed as random, rather than a fixed offset for each point. Point cloud refinement minimizes the cumulative effect of this error.

- **Camera Measurement Noise**: Camera measurements are influenced by resolution, lighting conditions, and sensor sensitivity. Low resolution or poor lighting reduces the accuracy of extracted features, impacting calibration quality. The error varies with environmental factors and is challenging to quantify directly.

- **IMU Measurement Noise**: IMU data, impacted by drift and noise, contributes directly to pose estimation and point cloud generation. Since IMU errors accumulate over time, they can propagate into calibration errors. The Xsens MTI-630R IMU introduces rotational and positional errors: 0.2° for roll and pitch, and 1° for yaw XSens (2024). However, since we rely on LiDAR-based odometry rather than dead reckoning alone, the IMU error's effect on final calibration is minimal.

- **Timing Errors**: Synchronization of each sensor's internal clock is crucial for accurate calibration. Although sensors are synchronized to minimize timing discrepancies, residual timing errors may still impact calibration. For instance, if the LiDAR clock (used for timestamps in odometry) is out of sync with the camera clock, this misalignment could introduce pose errors in depth image generation. Here, sensors are assumed to be correctly synchronized.

- **LiDAR to IMU Transform Error**: Since the odometry is referenced in the IMU frame, accurate estimation of the LiDAR-IMU transform is critical. This transform is obtained using the LiDAR_IMU_Init package, which introduces rotational and translational errors of $0.7244^\circ \pm 0.5076^\circ$ and 0.0133 ± 0.0102 meters, respectively Zhu et al. (2022b).

- **Fast-LIO-SLAM Errors**: Fast-LIO-SLAM introduces its uncertainties in odometry and point cloud generation. Minor errors arise from sensor variations and the SLAM algorithm's limitations. SC-PGO mitigates some of this drift, though specific improvements are undocumented. Fast-LIO-SLAM's reported drift is approximately

7 cm over 140 meters (or 0.05%) Kim (2024a). For a handheld distance of 110 meters and NTU eee_01 at 237 meters, the final drift is around 5.5 cm and 12 cm, respectively. SLAM post-processing reduces this drift further.

- **Motion-Induced Error**: Data acquired during motion may suffer from artifacts like motion blur in camera images and discrepancies in LiDAR scan alignment. Fast-LIO's IMU-based compensation addresses this, but any inaccuracies in IMU data could introduce residual errors.
- **SSIM Evaluation Error**: During calibration, SSIM optimization may not consistently yield ground-truth values, causing the optimizer to converge on suboptimal parameters. This is due to SSIM local minima within the parameter space, which might mislead the optimization, especially when adjusting one axis at a time. However, these values will be different for different datasets.
- **Odometry Error**: Depth image generation used spline interpolation to align the camera pose with LiDAR data capture. Fast-LIO-SLAM publishes poses at 10 Hz, with drone speeds averaging 0.70 m/s and peaking at 2.9 m/s, for the handheld combination it is a similar average speed with 0.89m/s and a max of 1.6 m/s. This results in an average distance between measurements begin 7 centimeters, or at most 30 centimeters, or 9 centimeters, and 16 centimeters for the Handheld combination. However, it is assumed the splines estimate the pose well and the difference between the actual pose and the pose resulting from the splines is small and negligible at these small distances.

5.2.1 Error Effect Analysis

There is an error and most likely this will always happen, but the effect of the error in the translation, rotation, Focal length, and Distortion needs to be determined. From equation 1 can be seen how the errors in the translation, rotation, and Focal length affect the making of the depth image and the place of the pixels. With equation 8 the effect of an error in the K1- distortion coefficient can be seen.
To further analyze the effects of these errors in the pointcloud, translation, rotation, focal length, and distortion coefficients mathematically, the equations are modified to include small errors in the respective parameters. These errors are denoted as Δ for each parameter. The intrinsic matrix K and the transformation matrix T are affected by errors in focal length (Δf) and extrinsics (ΔR, Δt). The modified equation becomes:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = (K + \Delta K) \cdot (T + \Delta T) \cdot \begin{bmatrix} x + \Delta P_x \\ y + \Delta P_y \\ z + \Delta P_z \\ 1 \end{bmatrix}. \tag{17}$$

Where:

$$\Delta K = \begin{bmatrix} \Delta f & 0 & 0 \\ 0 & \Delta f & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$T + \Delta T = \begin{bmatrix} R + \Delta R & t + \Delta t \\ 0 & 1 \end{bmatrix}.$$

Combining these gives:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f + \Delta f & 0 & c_x \\ 0 & f + \Delta f & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R + \Delta R & t + \Delta t \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x + \Delta P_x \\ y + \Delta P_y \\ z + \Delta P_z \\ 1 \end{bmatrix}. \tag{18}$$

For radial distortion, an error in the distortion coefficient k_1 is expressed as $k_1 + \Delta k_1$. The equations then become:

$$x_{\mathrm{d}} = x_{\mathrm{u}} \left(1 + (k_1 + \Delta k_1) r^2\right), \tag{19}$$

$$y_{\mathrm{d}} = y_{\mathrm{u}} \left(1 + (k_1 + \Delta k_1) r^2\right), \tag{20}$$

All of these equations can be combined further but this gives a complex long equation that is too hard to read and understand and will not be useful for this reason this is not done any further. But the effects of errors are as follows:

- **Pointcloud errors($\Delta P_x, \Delta P_y, \Delta P_z$):** are the errors in the points due to errors in the sensor data, or due to the pointcloud generation.
- **Translation errors ($\Delta t_x, \Delta t_y, \Delta t_z$) and Rotation errors ($\Delta R$):** Affects the matching of the location of the 3d point in the 2d image, leading to changes in u and v.
- **Focal length errors (Δf):** Introduce scaling effects, altering the pixel positions proportionally.
- **Distortion errors (Δk_1):** Affect the radial warping of the image, introducing non-linear pixel displacements.

5.3 Challenges

Throughout the project, there were several challenges, many coming from tasks taking longer than anticipated in the exploration and planning phases, but also from things just not working out as expected and needing to find a possible alternative.

5.3.1 Initial values

This method has as input a camera image, a pointcloud, odometry, and an initial set of values for the extrinsic and intrinsic parameters. From the results, it could be seen that the translation and the focal length could not accurately be calibrated, this resulted in only calibrating for the rotation and translation. This means the initial values for the translation and focal length won't be updated and need to be set as best as possible. For the Translation, this can be done by just measuring it with a ruler or a measuring tape, but for the focal length, this is more difficult. The best approach for this is to use the default parameters of the camera's datasheet if it is available.

5.3.2 Lack of Precise Ground Truth for Handheld Setup

The handheld configuration does not have precise ground truth data, making it challenging to determine the quality of the calibration. The checkerboard calibration was done, here the intrinsic parameters were determined well, but the extrinsic contained some errors, especially in the translation. This makes it difficult to determine how big the error in the extrinsic calibration for the handheld combination is. The other datasets were chosen because they have good ground truth values with which the error in the calibration can be determined.

5.3.3 Time Management

The initial plan was to perform both intrinsic and extrinsic calibration with a neural network, needing depth images derived from point clouds. Developing the code to make these images took longer than expected, in addition to unforeseen bugs. Additionally, switching

from the initial sensors setup to a handheld setup introduced further complexities, as the new LiDAR model required adjustments in the point cloud generation software due to the initial setup's different LiDAR. This recalibration method ultimately shifted toward using a Scipy optimizer with a structural similarity index (SSIM)-based loss function. Though depth image generation is still necessary, this process—originally planned as part of preprocessing alone—lacks GPU acceleration, slowing calibration due to the computational demands of CPU-only depth image processing. The initial neural network approach is further detailed in section 5.6.2 below.

5.3.4 Generalization to Other Scenarios

This method gave similar results to all the datasets used, this can be seen from table 5 and table 6. While the results are the same, the evaluation between the datasets is different, especially for the Kitti dataset. But these are just a small set of all possible environments. From earlier testing with an older version of the handheld combination worse pointcloud and images were recorded, this resulted in a bad calibration, so this approach does need to be tested more to see how it reacts to different environments. For a better understanding or validation if this approach works for other situations would be to do the calibration for the same sensor setup for different environments. This is done with the NTU sensor system where 2 datasets are used, While the calibration results are similar, the graph of the SSIM evaluation for one DOF is different, especially for the rotation. Where in the sbs Roll there are multiple peaks instead of one peak as in the eee dataset. This indicates that the SSIM function needs to use different parameters for the calibration like window size and possibly the weight between the SSIM terms. Another test that should be done is to do the calibration for the same environment, but different sensor systems to see the effect of different sensors, and placements on the approach. While all of these are good, first the accuracy of the approach should be done to be able to analyze the effect of the environment and sensors.

5.4 Strengths of the Approach

This calibration approach, if it works as expected, removes the need for a lab environment with predefined calibration targets or several static features, which are typically required for reliable results. Instead, it is doable in diverse environments, allowing calibration with arbitrary numbers of cameras and LiDARs, making it robust for various configurations without extensive environmental setup.
Another strength of this approach would be its ability to also estimate intrinsic parameters, which sets it apart from many other automatic calibration methods. Most existing methods still rely on a controlled lab environment with specialized equipment to determine these parameters, such as the focal length, principal point, and distortion, which can be both time-consuming and resource-intensive. However, this method removes the need for such constraints, giving a more flexible and practical alternative.
But with the current accuracy of this approach, it to be used for an initial estimation of the parameters, which then can be used by other approaches for a more accurate calibration.

5.5 Current Limitations of the Approach

The approach's accuracy relies on having the same structures and changes in both the images of the different modalities. Certain image features, such as shadows, wall textures, and road markings, which are visible only to camera sensors, can reduce calibration accuracy because these are not seen in the depth image. This method also relies on IMU data to generate odometry and point clouds, which, while useful, could introduce errors if the IMU

is removed or reduces alignment, affecting both odometry and point cloud quality and reducing the calibration accuracy.
Another limitation of this approach is the significant time required to perform the calibration process. As can be seen in the results section, calibrating a single image pair can take over 50 seconds. This means for a dataset of 50 images, the total calibration time exceeds 40 minutes, which is a lot of time. With a GPU this would be able to be done instantly. This long processing time makes the approach impractical for real-time applications, where calibration would need to occur multiple times per second or at least once every second to keep up with dynamic environments or changing conditions.
Further limitations are that the SSIM function does not have its peaks at the ground truth values, the quality of the LiDAR data and the effect of the environment, the quality of the images, and the effect of the lighting and how that results.

5.5.1 SSIM Peaks/valleys

The valleys or peaks for the minimizer are not (always) at an adjustment of 0, but at a value slightly to one of the sides, these peaks are also not all at the same adjustment value for the different window sizes. Some of the possible reasons for this are that the ground truth value is not perfect and the found adjustments are the ground truth, but looking at the depth images made with peak values these seem not correct. The input data, the camera image, or the lidar data with imu, contain noise or errors that result in the peak adjustment giving a lower SSIM value. The possible LiDAR/IMU error is then a more likely reason, these are used in other processes, that can add errors, to make the pointcloud and the odometry and this results in a wrong depth image. Another possibility is multiple similar features in the depth and/or camera image that are "matched" wrongly, the environment the NTU dataset is recorded in, and the building is very similar when a small change in translation is done the building still looks the same. These peak values are used to make a depth image and rectify the camera image, these can be seen in figures 19, 44, and 53. Inspecting these images, camera and depth images made with the ground truth values match the best, this means the error is not in the ground truth values. All of these other possibilities together with issues specific to each dataset, environment, effects of different window sizes, and noise have combined that the peak values are slightly off and not at an adjustment of zero.

5.5.2 LiDAR Data quality & environment size

The accuracy of the calibration process is significantly affected by the quality of the data and the depth field of the environment in the images, as both impact the resolution and comparability of LiDAR and camera data. For LiDAR, the use of all available data to create a dense point cloud reduces the part of the issues. However, this approach cannot be applied to camera images, where pixel size plays a critical role.
Each camera and depth image pixel corresponds to a specific area in the environment. When the environment is far from the sensors, each pixel represents a larger area, making finer details harder to resolve or not visible. Conversely, in closer or smaller environments, pixels represent smaller areas, allowing for greater detail. This relationship means that translation changes result in larger pixel shifts in smaller environments than in larger ones. Similarly, small rotations cause more noticeable changes in smaller environments compared to larger ones.
The same principle applies to intrinsic parameters such as distortion and focal length. When changes in these parameters result in changes smaller than the pixel size, the calibration process struggles to optimize them effectively. This limitation can be seen in datasets like NTU and KITTI, which are captured in large outdoor environments. The handheld dataset, while taken indoors, also partly has the same issue, the long hallways of the building do

affect it together with the lower pointcloud and image quality. The coarse granularity of pixel correspondence in such environments may contribute to lower calibration accuracy compared to other methods.
Overall, the accuracy of this calibration approach depends on the ability to capture the environment at a high resolution, where each pixel corresponds to a small area. Improved image and data quality are essential for achieving better calibration results.

5.5.3 Image quality and lighting conditions

In addition to LiDAR data quality, the accuracy of calibration depends heavily on image quality and the lighting conditions in the environment. As discussed in Section 3.5.1, it is expected that pixel intensity decreases with increasing distance due to reduced light reaching the camera sensor. However, this assumption does not always hold, especially in scenarios with significant external light sources, such as sunlight or artificial lighting. Reflections from bright surfaces, such as white walls or garage doors (common in the KITTI dataset), further complicate this relationship.
The approach was tested and maybe optimized for the NTU dataset which was recorded in an area, where at the time of recording, there was less sunlight or reflecting surfaces that affected the accuracy. This is one of the reasons the accuracy is better on the NTU dataset than on the Kitti or the Handheld one.
Lighting challenges also affected the ground truth calibration for the handheld dataset. Of the 4800 images captured, only in 300 images the checkerboard pattern was detected, despite the checkerboard being present in most of the images. Issues like image blur and poor corner detection caused by lighting variations significantly reduced the usable data.
These issues negatively impacted calibration accuracy. Future efforts could be improved by preprocessing camera images to minimize these lighting effects. Techniques such as denoising, histogram equalization, or adaptive thresholding could enhance image quality and improve the robustness of calibration under varying lighting conditions. This preprocessing step would be particularly valuable in scenarios with reflections and inconsistent lighting, ultimately leading to better calibration accuracy.

5.6 Potential Improvements & future work

Future improvements are to make the approach work, this can be done by using different optimizers, using different parameters in the SSIM function that give a better gradient, or maybe even using a different function to compare the images to SSIM. One of these other functions is to use a neural network to determine the similarities between the two images, but for this, a whole neural network needs to be trained and tested for different environments. But then the better option is to remove the optimizer and make the neural network also do that if it is possible. Another future work is to make the depth image generation faster and work with the GPU, to reduce the load on the CPU and instantly make the depth images. With a faster calibration, more things could be tested, and even make the model work in a real-time application if the calibration accuracy also improves.
Another improvement is to slowly increase the number of degrees of freedom for the optimizer, with one degree of freedom it can be optimized with a small error which could be seen from the SSIM evaluation. Then increase from one DOF to two DOF, for example only x and y translation, or only yaw and either the roll or the pitch, and see how that affects the calibration. From there on a DOF can be added to see if it still works. While this may still not work, this could identify the current problem with the calibration better.

5.6.1 Parameter changes per environment

Currently, the same parameters are used for all environments, but the approach could be refined by using specific parameters for specific environmental conditions. For instance, the optimal window size in the SSIM function might vary depending on the relationship between pixel size and the size of features in the environment.
In environments where each pixel corresponds to a small area, but the features within that area are relatively large, a larger window size would be better to capture enough information. On the other hand, in environments with smaller features, a smaller window size might be more effective to ensure finer details are preserved.
Another parameter worth adjusting is the weighting between the components of the SSIM function. Currently, the luminance term is set to 0, with both contrast and structure terms weighted equally at 1. Exploring different weightings for these components could reveal combinations better suited for specific environments or camera resolutions. For example, in high-resolution setups, emphasizing structure might yield better results, whereas environments with challenging lighting conditions could benefit from incorporating the luminance term to some extent.
Changing these parameters based on environmental characteristics and sensor resolution could significantly improve the accuracy and robustness of the calibration process.

5.6.2 Possible Neural Network approach

A potential improvement in the calibration process could involve using a neural network for both the extrinsic and intrinsic calibration. In Figure 23 a possible model structure is shown, where the inputs would be the camera images and their corresponding depth images. The training process could include augmenting the camera images with distortion and generating depth images with variations in rotation, translation, and focal length. These images could then pass through a ResNet model separately to extract features, with the outputs pooled and concatenated before being processed through dense layers to regress calibration parameters. Adding time-distributed layers might help the model use sequential images for more accurate calibration estimates.
An alternative approach might combine iterative optimization for the initial parameters and the neural model for the finer detail, given the difficulty of accurately estimating the parameters. This method could compare depth images with camera images using metrics such as SSIM, which could be integrated into the neural network's loss function. By using the camera images based on the network's estimated focal length and distortion values, the network could aim to minimize the difference with the depth images. For future development, the neural network approach would require substantial training data and an exploration of different network architectures to prevent overfitting. Additional modifications, such as improved loss functions or advanced feature extraction techniques, could make this method more viable.

5.7 Comparison to other Approaches

In the literature, several approaches exist for calibration, ranging from traditional methods to neural network-based techniques. Some differ significantly in their approach, while others only vary in aspects such as loss functions or the architecture of neural networks.
This approach shares similarities with CaLiCaNet (Rachman et al., 2023), as both use depth images generated from LiDAR data and corresponding camera images to estimate intrinsic and extrinsic calibration parameters. However, a key difference lies in the methodology: while CaLiCaNet uses a neural network to estimate these parameters, this approach uses a structural similarity index SSIM-based optimization. Furthermore, unlike CaLiCaNet, which generates depth images from a single LiDAR frame, this approach combines multiple

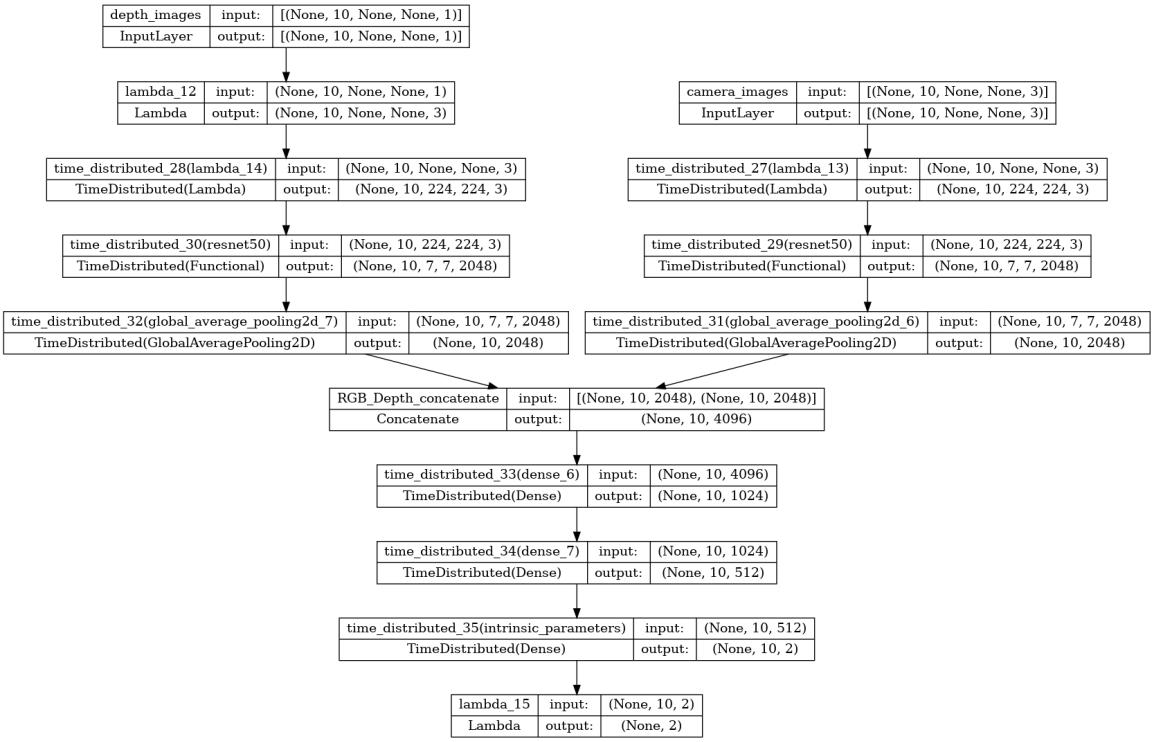

Figure 23: Neural Network model structure

LiDAR frames into a dense pointcloud, resulting in a higher-quality depth image. This change could improve the data quality and potentially improve calibration robustness if there is minimal or no error in the generation.

In contrast, INF (Zhou et al., 2023) uses a fundamentally different strategy by constructing neural density fields for LiDAR data and neural color fields for camera images and then comparing these 3D representations. Essentially, INF emphasizes creating 3D models of the environment based on sensor data and aligning these models. This contrasts sharply with the 2D approach taken in this approach, where depth images and camera images are compared directly in the image plane. By focusing on 2D comparisons, this approach simplifies the computational requirements while maintaining flexibility in adapting to diverse environments.

Overall, this approach diverges from others by emphasizing dense pointcloud generation and 2D image comparisons, offering a simpler yet effective alternative to 3D model-based or neural network-heavy methods. While accuracy comparisons are outside the scope of this discussion, the methodological differences highlight the potential of this approach to provide a balance between complexity and calibration performance.

6 Conclusion

In this thesis, we study the feasibility of using SSIM for a targetless auto-calibration of a Camera intrinsic and extrinsic of any number of LiDAR and camera combinations. By removing the need for targets, this approach would be able to be done anywhere and on already recorded data. The approach uses the LiDAR scans and combines them to make a Pointcloud and Odometry with the use of the IMU. From this pointcloud Odometry depth images can be made to be compared to the camera images, to optimize the extrinsic and intrinsic parameters to find the depth image that has the best comparison. The comparison is done by using an adapted Structural Similarity Index Measure that compares the contrast and structure of the two images in a specific window, the Luminance term is not used here because of the different image modalities.
The results show that while the targetless approach can work with some adjustments, the accuracy of the approach is not good, and is not able to calibrate the translation with small miss calibrations. The results are not comparable to the state-of-the-art methods that can calibrate not only the rotation but also the translation, or even Intrinsic parameters, and both with better accuracy. The SSIM evaluation shows that, at least for the NTU eee data, small rotational adjustments change the SSIM value much more than small translation adjustments, and the intrinsic calibrations are harder to optimize. The SSIM evaluation on the NTU and other datasets seems to indicate that the calibration is possible for the rotation, but for all dataset it indicates it is not possible to do the calibration for the translation and focal length.
The approach developed in this thesis was benchmarked to other methods, CaLiCaNet, CalibNet, and CalibRCNN. These methods outperformed our approach but a large amount, our approach is better for an initial estimate which then can be used for one of the state-of-the-art approaches. The result highlighted that the SSIM function needs some adjustments to get better calibration or a combination of both SSIM and a neural network where SSIM is used in the loss function may help the training and get better accuracy.
Future work should focus on improving the accuracy of the calibration for the rotation and distortion and adding the translation and focal length in the calibration. Possibly with a combined optimization rather than multiple separate optimizations, possibly with bounds to make the search space smaller. A neural network approach for the smaller adjustments and letting SSIM do the initial calibration. Anohter posibility is combining multiple SSIM window sizes for bigger and finer adjustments. Future work also needs to change the method used to make the depth images, since the current method uses functions that do not support GPU acceleration, and depth image generation now takes too long. With this speed, a possible online approach is not possible since the result will be way too late.
In conclusion, our approach shows a potential method to automatically calibrate the extrinsic and intrinsic parameters between a LiDAR and Camera pair, without the use of targets that at this time does not work. There is a lot of room for improvement but a good foundation is set and further development for better accuracy, precision, fast calibration, and more parameters.

References

J. P. Barreto and H. Araujo. Geometric properties of central catadioptric line images and their application in calibration. *IEEE transactions on pattern analysis and machine intelligence*, 27:1327–33, 09 2005. doi: 10.1109/TPAMI.2005.163.

O. Bogdan, V. Eckstein, F. Rameau, and J.-C. Bazin. Deepcalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view cameras. pages 1–10, 12 2018. doi: 10.1145/3278471.3278479.

D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997. doi: 10.1109/CVPR.1997.609468.

Hesai Technology. Hesai pandar xt-32 brochure, 2021. URL `https://www.oxts.com/wp-content/uploads/2021/01/Hesai-PandarXT32_Brochure.pdf`. Accessed: 2024-06-18.

Intel Realsense. Intel realsense d400 series product family, 1 2019. URL `https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf`. Accessed: 2024-06-19.

G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna. Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018. doi: 10.1109/iros.2018.8593693. URL `http://dx.doi.org/10.1109/IROS.2018.8593693`.

G. Kim. Fast_lio_slam: Fast lidar-inertial odometry and mapping with slam capability, 2024a. URL `https://github.com/gisbi-kim/FAST_LIO_SLAM`. Accessed: 2024-9-17.

G. Kim. Sc-a-loam: Scan context-aware lidar odometry and mapping, 2024b. URL `https://github.com/gisbi-kim/SC-A-LOAM`. Accessed: 2024-9-17.

J. Kümmerle and T. Kühner. Unified intrinsic and extrinsic camera and lidar calibration under uncertainties. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6028–6034, 2020. doi: 10.1109/ICRA40945.2020.9197496.

V. V. Lehtola, H. Kaartinen, A. Nüchter, R. Kaijaluoto, A. Kukko, P. Litkey, E. Honkavaara, T. Rosnell, M. T. Vaaja, J.-P. Virtanen, et al. Comparison of the selected state-of-the-art 3d indoor scanning and point cloud generation methods. *Remote sensing*, 9(8):796, 2017.

J. Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems*, 2013. URL `https://api.semanticscholar.org/CorpusID:10004324`.

H. li, R. Hartley, and J.-H. Kim. A linear approach to motion estimation using generalized camera models. 06 2008. doi: 10.1109/CVPR.2008.4587545.

L. Li, H. Li, X. Liu, D. He, Z. Miao, F. Kong, R. Li, Z. Liu, and F. Zhang. Joint intrinsic and extrinsic lidar-camera calibration in targetless environments using plane-constrained bundle adjustment, 2023. URL https://arxiv.org/abs/2308.12629.

T. Ma, Z. Liu, G. Yan, and Y. Li. Crlf: Automatic calibration and refinement based on line feature for lidar and camera in road scenes, 2021.

Mathworks. URL https://nl.mathworks.com/help/lidar/ug/lidar-camera-calibration.html.

T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie. Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. *The International Journal of Robotics Research*, 41(3):270–280, 2022.

A. Rachman, J. Seiler, and A. Kaup. End-to-end lidar-camera self-calibration for autonomous vehicles, 2023. URL https://arxiv.org/abs/2304.12412.

J. Shi, Z. Zhu, J. Zhang, R. Liu, Z. Wang, S. Chen, and H. Liu. Calibrcnn: Calibrating camera and lidar by recurrent convolutional neural network and geometric constraints. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10197–10202, 2020. doi: 10.1109/IROS45743.2020.9341147.

R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987. doi: 10.1109/JRA.1987.1087109.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2. URL https://scipy.org/.

Vision. Sensing breakdown: Waymo jaguar i-pace robotaxi. https://www.tangramvision.com/blog/sensing-breakdown-waymo-jaguar-i-pace-robotaxi#calibrate-all-the-sensors, 2021. Accessed: 2024-10-07.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

XSens. Mti 600-series datasheet, 2024. URL https://www.xsens.com/hubfs/Downloads/Leaflets/MTi%20600-series%20Datasheet.pdf. Accessed: 2024-06-19.

W. Xu and F. Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter, 2021. URL https://arxiv.org/abs/2010.08196.

G. Yan, F. He, C. Shi, P. Wei, X. Cai, and Y. Li. Joint camera intrinsic and lidar-camera extrinsic calibration. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11446–11452, 2023. doi: 10.1109/ICRA48891.2023.10160542.

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

S. Zhou, S. Xie, R. Ishikawa, K. Sakurada, M. Onishi, and T. Oishi. Inf: Implicit neural fusion for lidar and camera. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10918–10925, 2023. doi: 10.1109/IROS55552.2023.10341648.

F. Zhu, Y. Ren, and F. Zhang. Robust real-time lidar-inertial initialization. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955. IEEE, 2022a.

F. Zhu, Y. Ren, and F. Zhang. Robust real-time lidar-inertial initialization, 2022b. URL `https://arxiv.org/abs/2202.11006`.

A Dataset Example images

A.1 Handheld combination Images

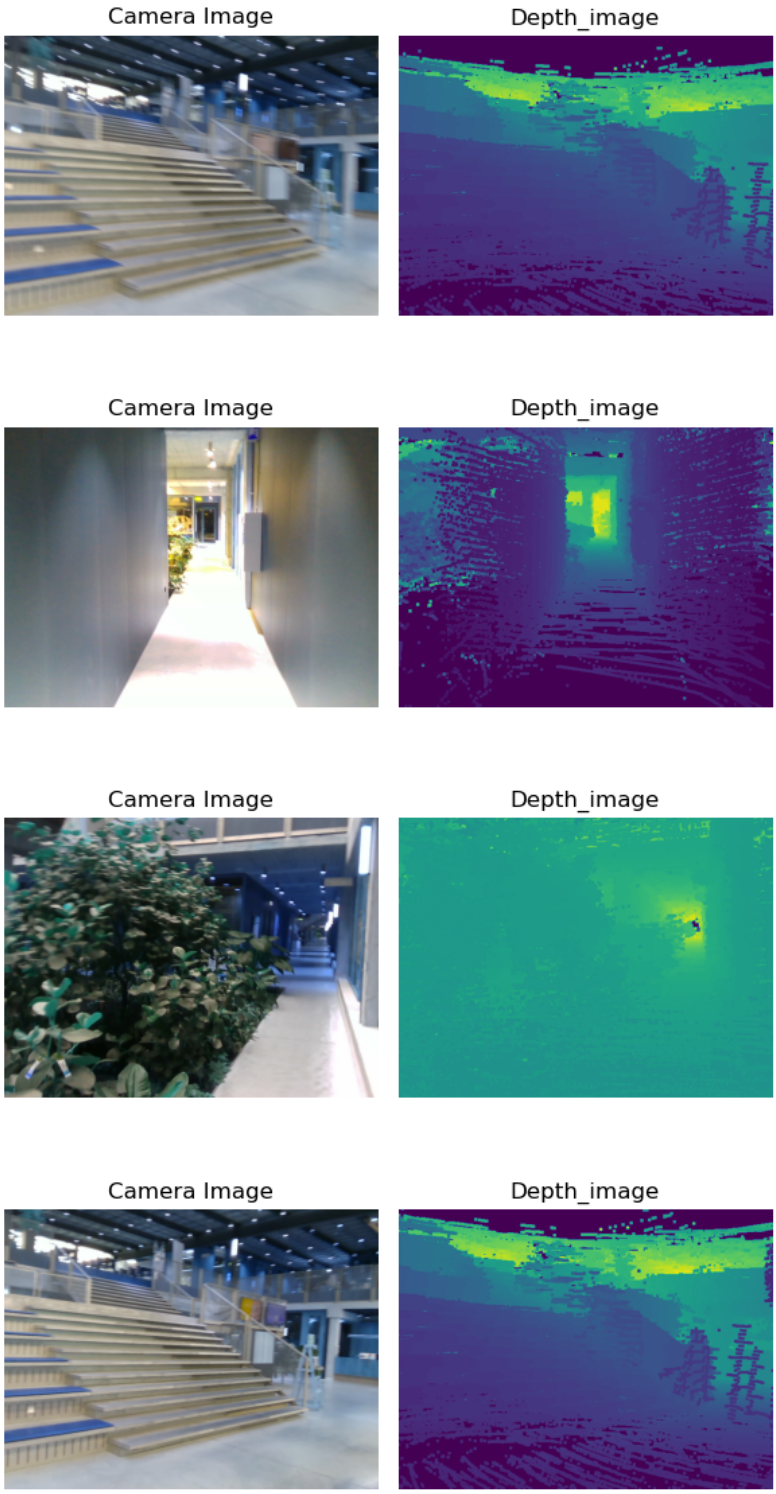

Figure 24: Example camera and depth images of the Handheld combination bag 3

A.2 NTU eee_01 Images

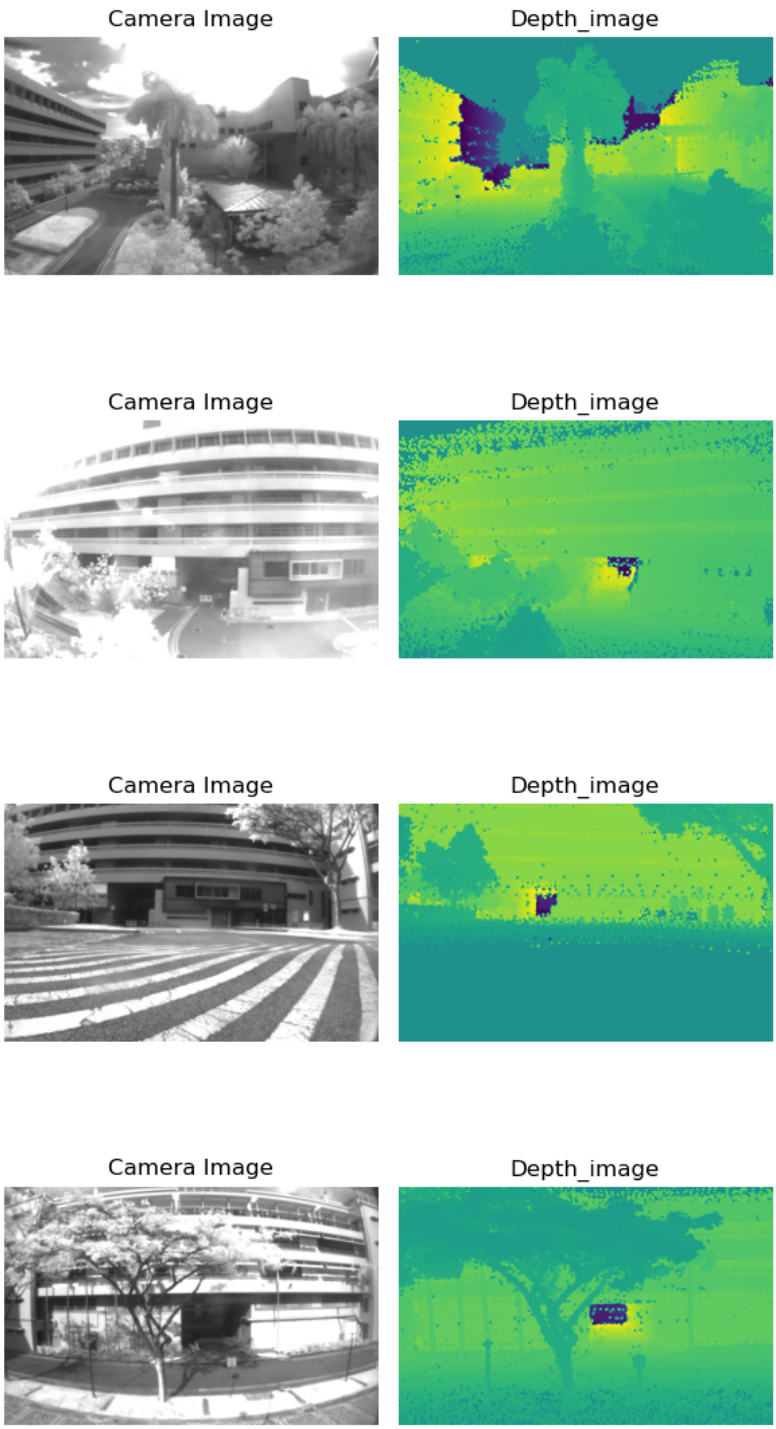

Figure 25: Example camera and depth images of the NTU eee_01

A.3 NTU sbs_01 Images

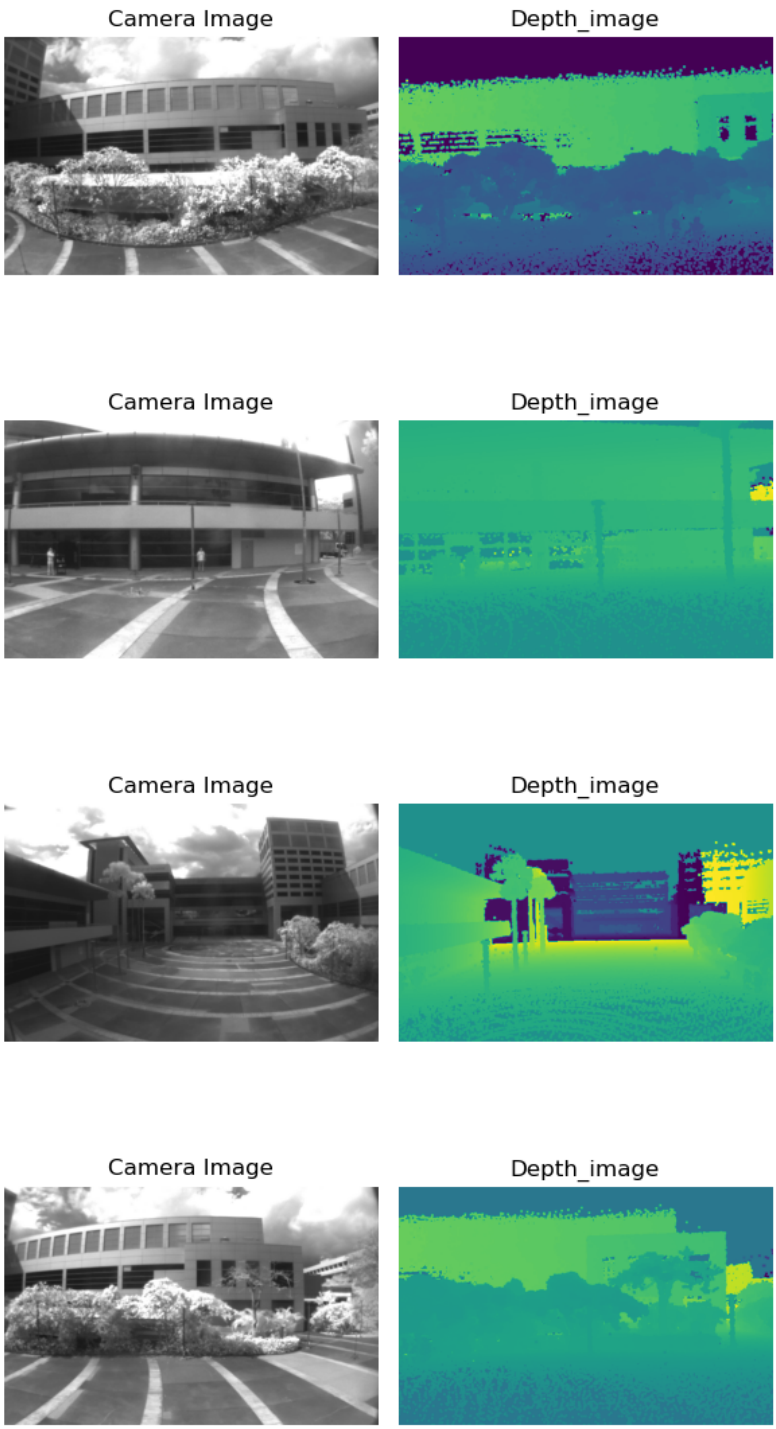

Figure 26: Example camera and depth images of the NTU sbs_01

A.4 Kitti Images

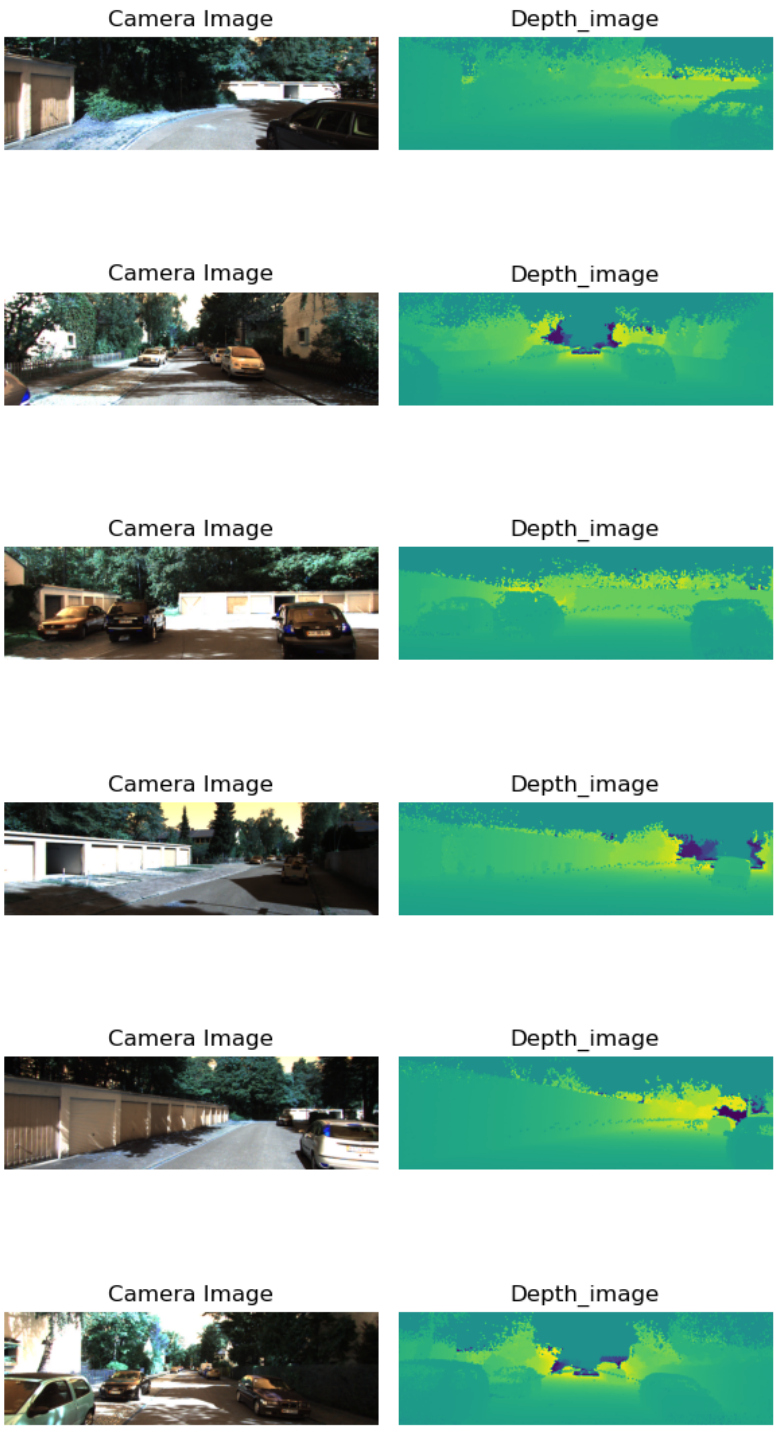

Figure 27: Example camera and depth images of the Kitti 2011_09_26_drive_0022

B SSIM evaluation Result plots

B.1 NTU SBS_01 SSIM Eval plots

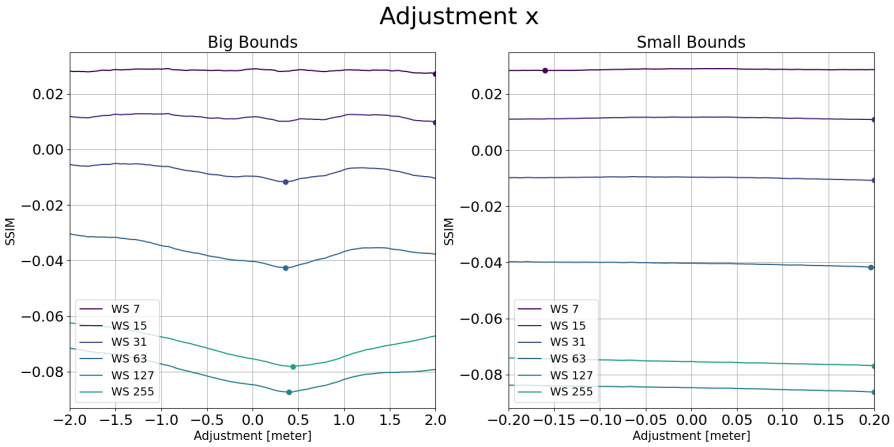

Figure 28: NTU SBS_01 SSIM Evaluation X-axis

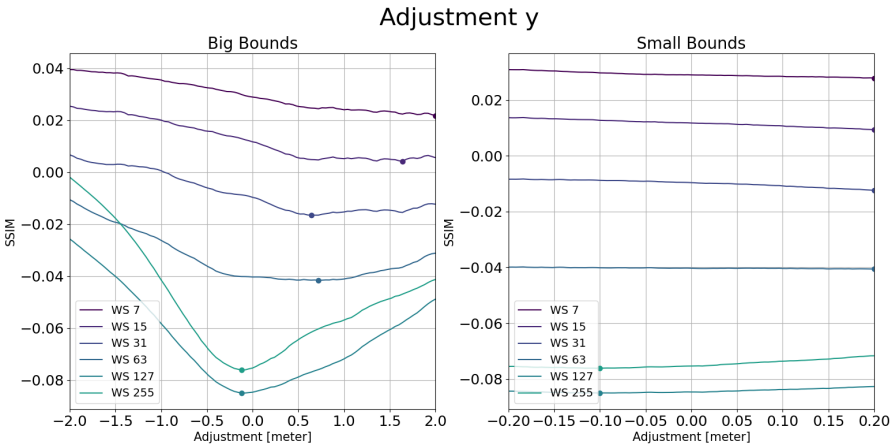

Figure 29: NTU SBS_01 SSIM Evaluation Y-axis

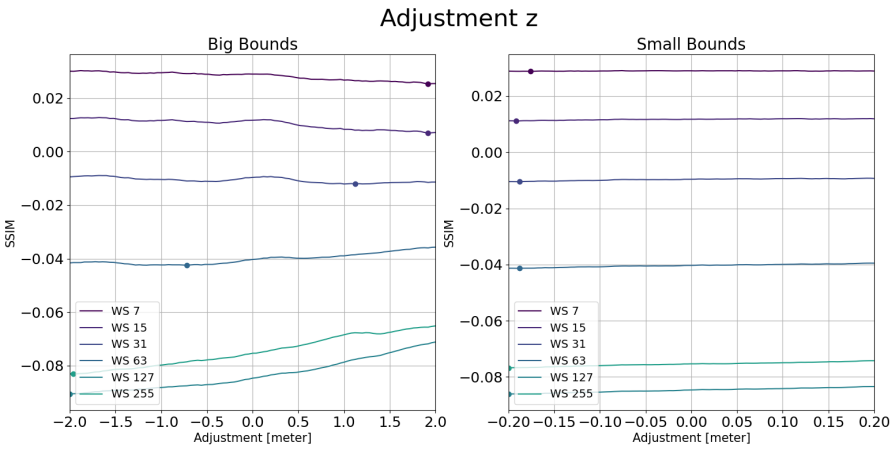

Figure 30: NTU SBS_01 SSIM Evaluation Z-axis

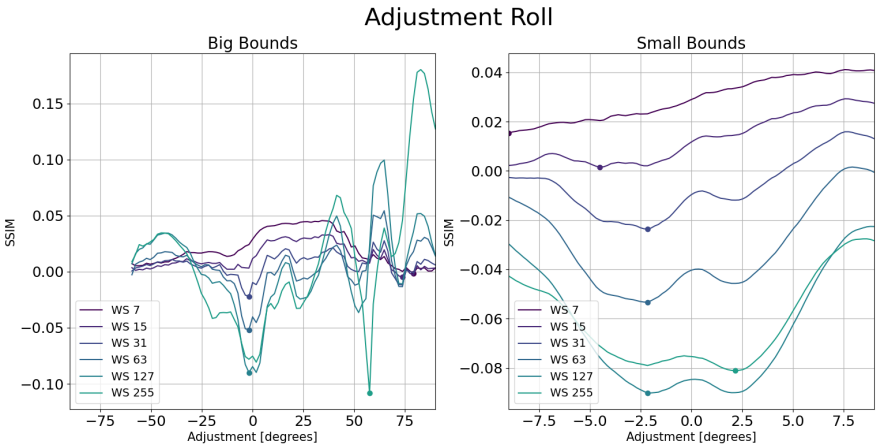

Figure 31: NTU SBS_01 SSIM Evaluation Roll

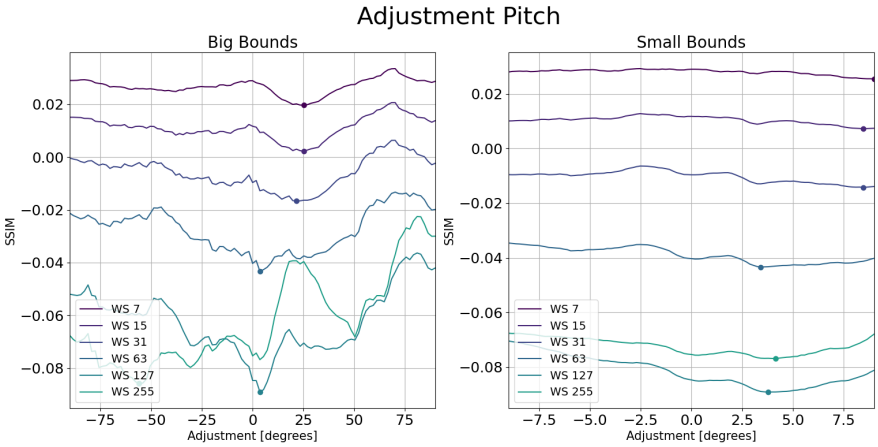

Figure 32: NTU SBS_01 SSIM Evaluation Pitch

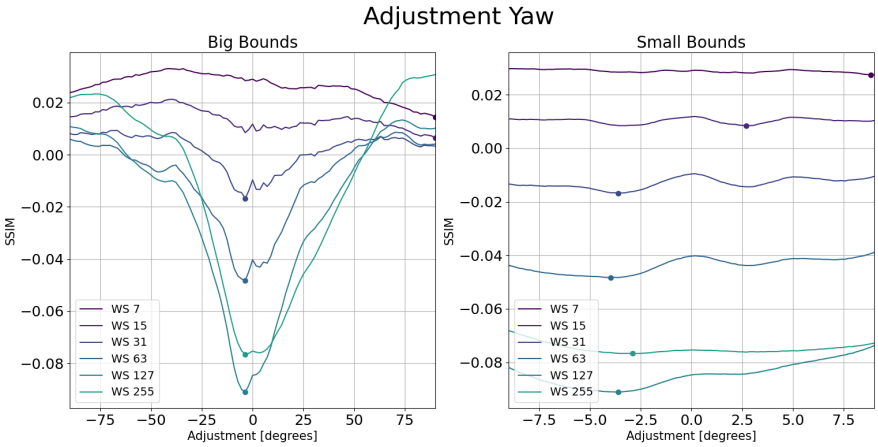

Figure 33: NTU SBS_01 SSIM Evaluation Yaw

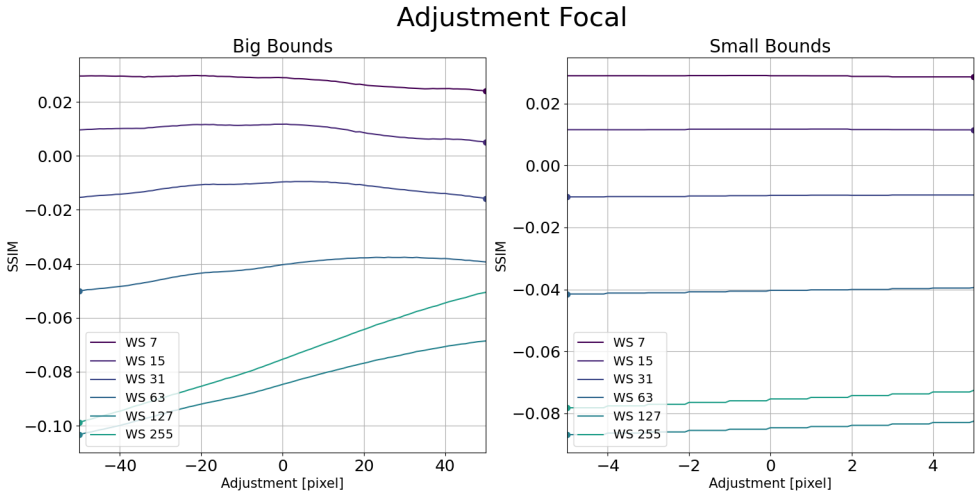

Figure 34: NTU SBS_01 dataset SSIM Evaluation Focal length

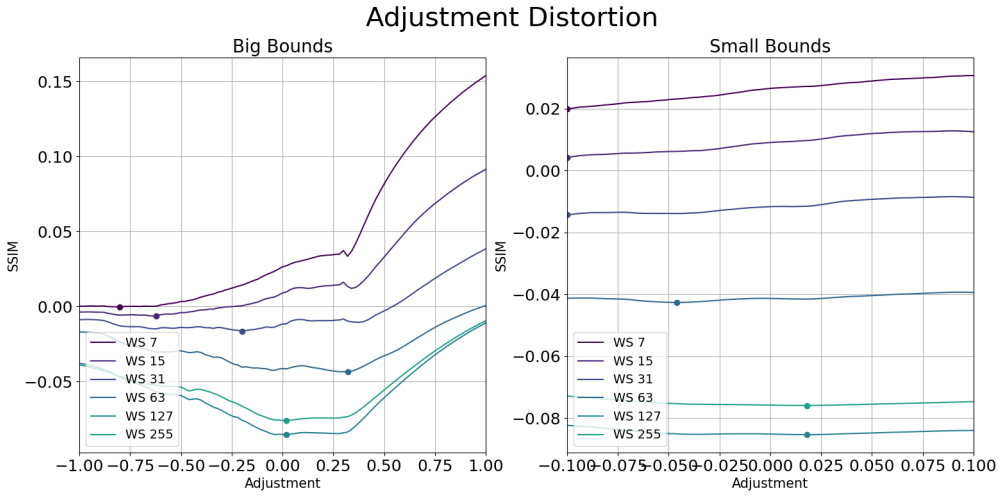

Figure 35: NTU SBS_01 dataset SSIM Evaluation Distortion

Table 9: Error value for one degree of freedom SSIM-calibration for the NTU SBS_01 combination dataset, the error here is the absolute difference between the peak value and ground truth value.

Window Size [pixel]	x [m]	y [m]	z [m]	Roll [°]	Pitch [°]	Yaw [°]	Focal [pixel]	Distortion
7	2.00	2.00	1.92	79.20	25.20	90.00	50.00	0.80
15	2.00	1.64	1.92	73.80	25.20	90.00	50.00	0.62
31	**0.36**	0.64	1.12	**2.16**	21.60	3.60	50.00	0.20
63	**0.36**	0.72	**0.72**	**2.16**	**3.42**	3.96	50.00	0.32
127	0.40	**0.10**	2.00	**2.16**	3.78	3.60	50.00	**0.02**
255	0.44	**0.10**	1.96	57.60	55.80	**2.88**	50.00	**0.02**

B.2 Kitti SSIM Eval plots

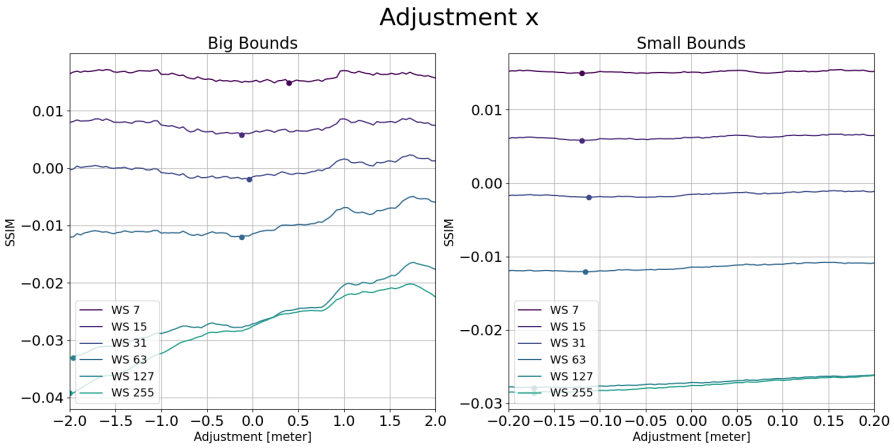

Figure 36: Kitti SSIM Evaluation X-axis

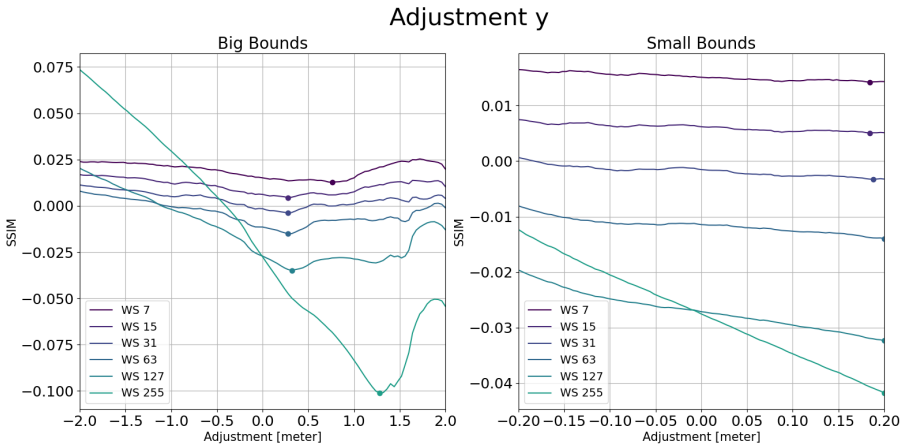

Figure 37: Kitti dataset SSIM Evaluation Y-axis

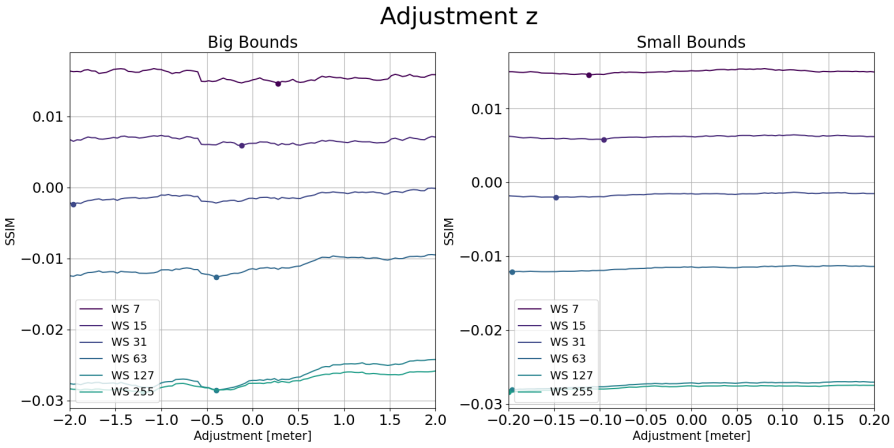

Figure 38: Kitti dataset SSIM Evaluation Z-axis

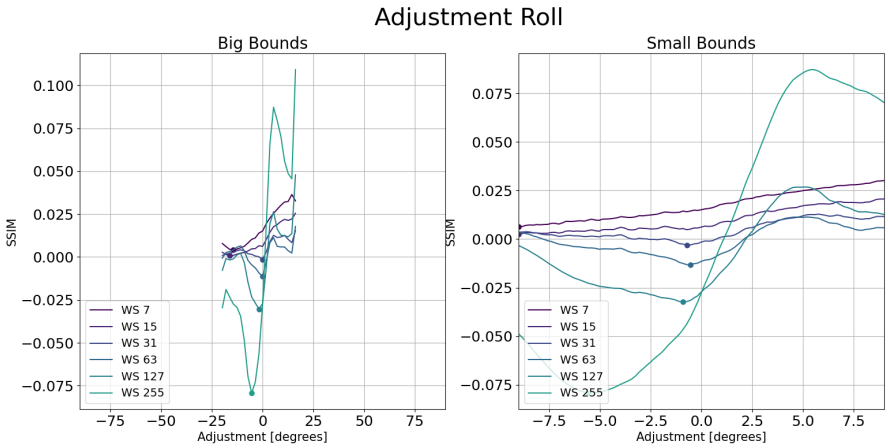

Figure 39: Kitti dataset SSIM Evaluation Roll

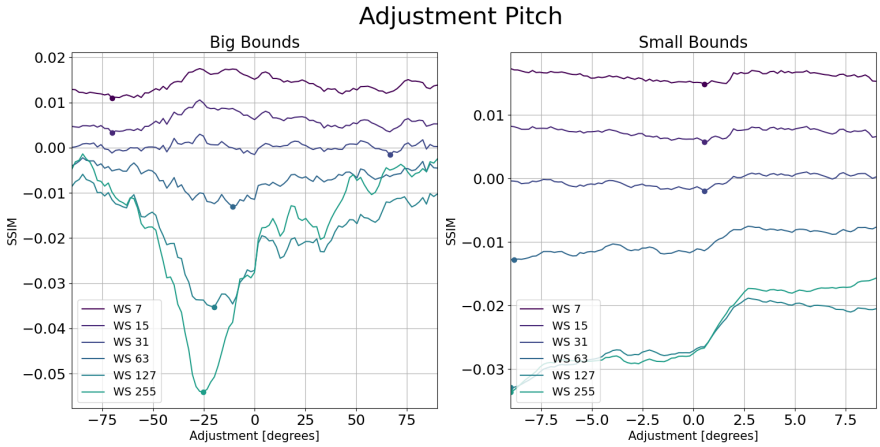

Figure 40: Kitti dataset SSIM Evaluation Pitch

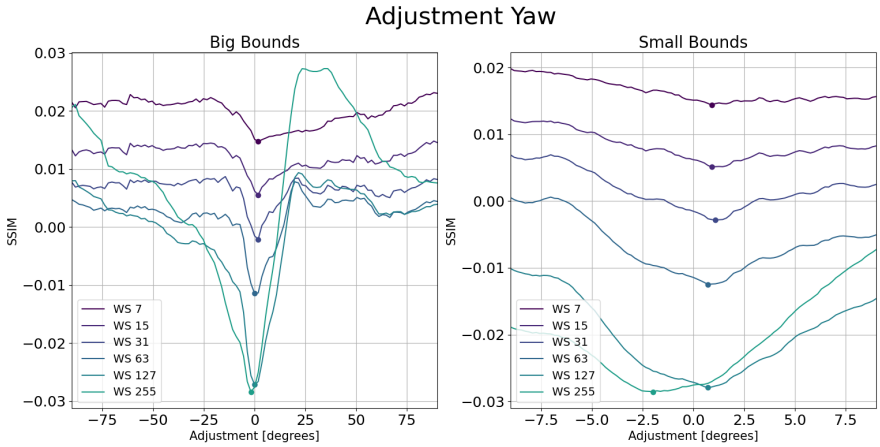

Figure 41: Kitti dataset SSIM Evaluation Yaw

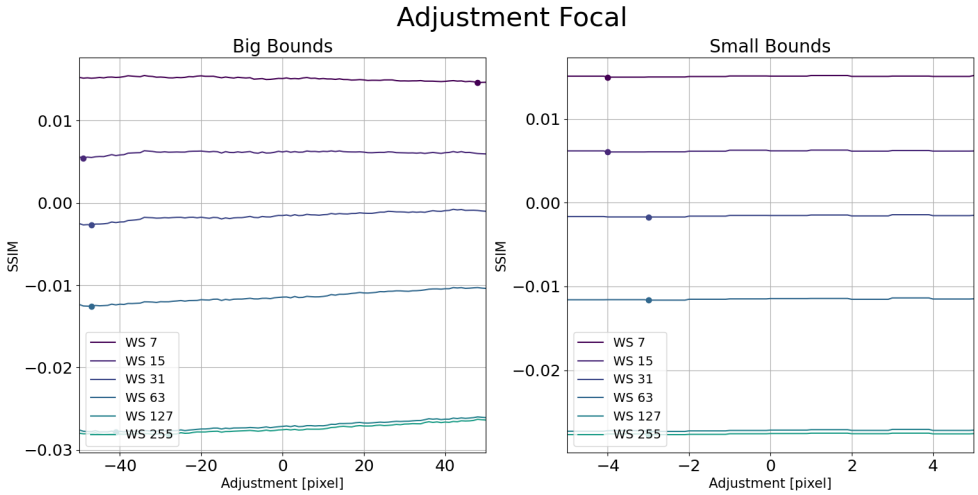

Figure 42: Kitti dataset SSIM Evaluation Focal length

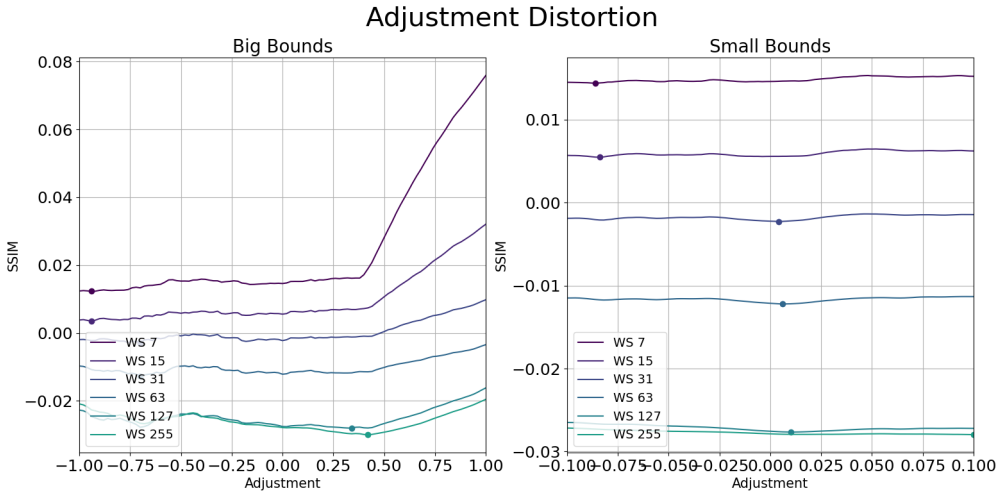

Figure 43: Kitti dataset SSIM Evaluation Distortion

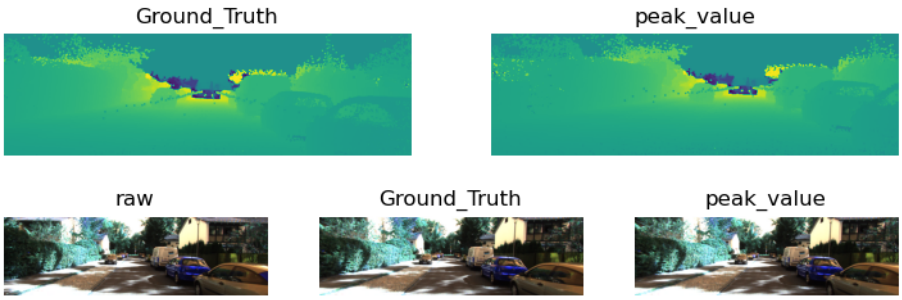

Figure 44: Kitti ground truth and optimal depth and camera images.

Table 10: Error value for one degree of freedom SSIM-calibration for the Kitti dataset, the errors here is the absolute difference between peak value and ground truth value.

Window Size [pixel]	x [m]	y [m]	z [m]	Roll [°]	Pitch [°]	Yaw [°]	Focal [pixel]	Distortion
7	0.40	0.76	0.11	14.40	70.20	0.90	48.00	0.94
15	0.12	**0.28**	**0.10**	16.20	70.20	0.90	49.00	0.94
31	**0.11**	**0.28**	1.96	0.72	**0.54**	1.08	47.00	0.70
63	0.12	**0.28**	0.40	**0.54**	10.80	**0.72**	47.00	0.70
127	1.96	0.32	0.40	0.90	19.80	**0.72**	41.00	**0.34**
255	2.00	1.28	1.20	5.58	25.20	1.98	**28.00**	0.42

B.3 Handheld SSIM Eval plots

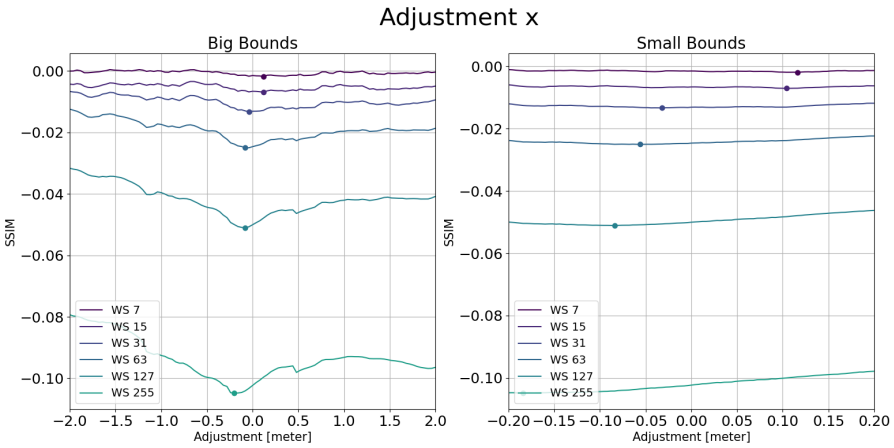

Figure 45: Handheld SSIM Evaluation X-axis

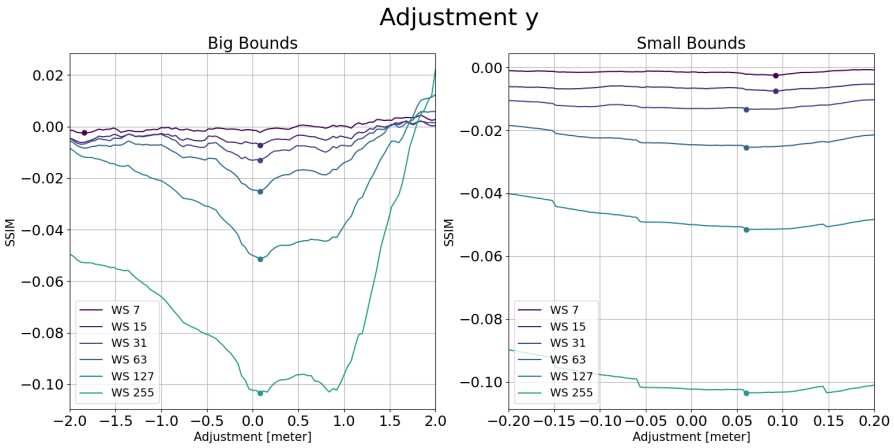

Figure 46: Handheld SSIM Evaluation Y-axis

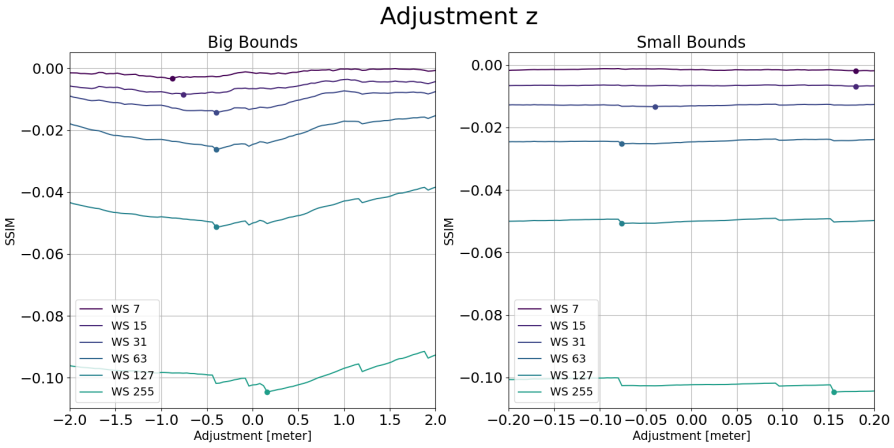

Figure 47: Handheld SSIM Evaluation Z-axis

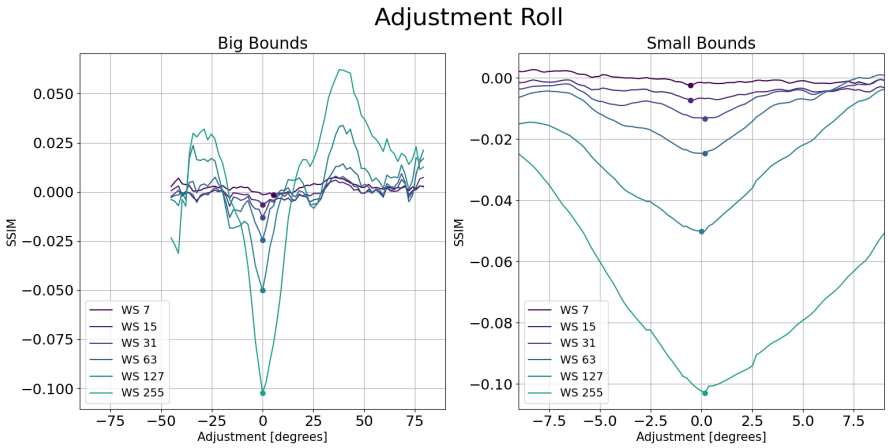

Figure 48: Handheld SSIM Evaluation Roll

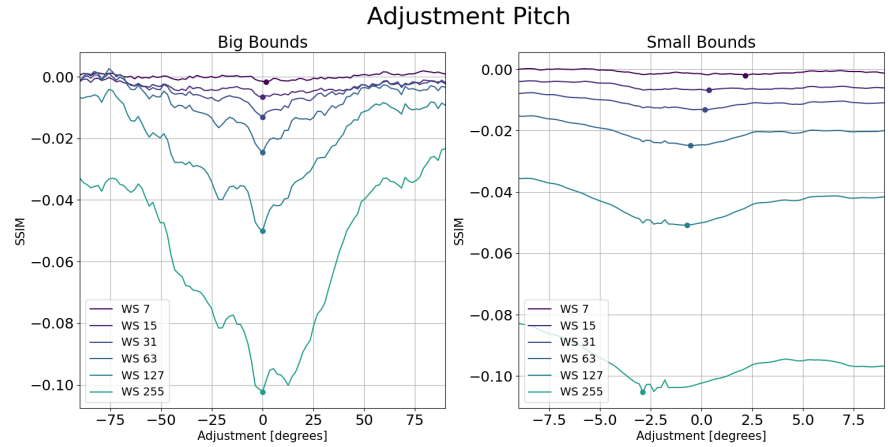

Figure 49: Handheld SSIM Evaluation Pitch

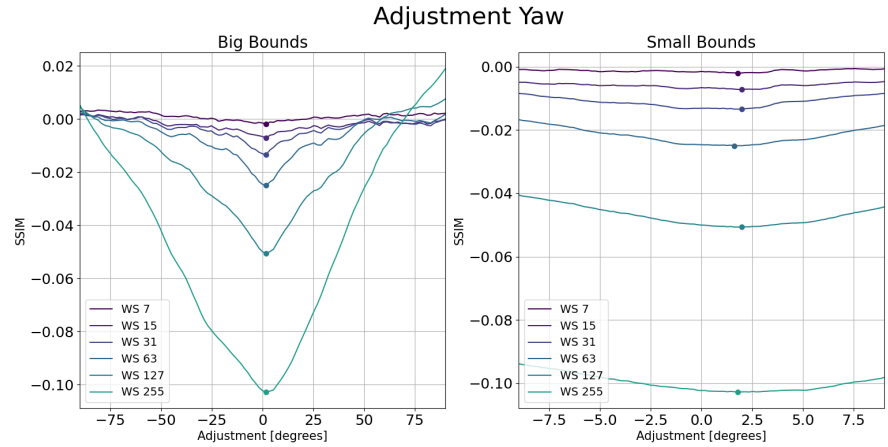

Figure 50: Handheld SSIM Evaluation Yaw

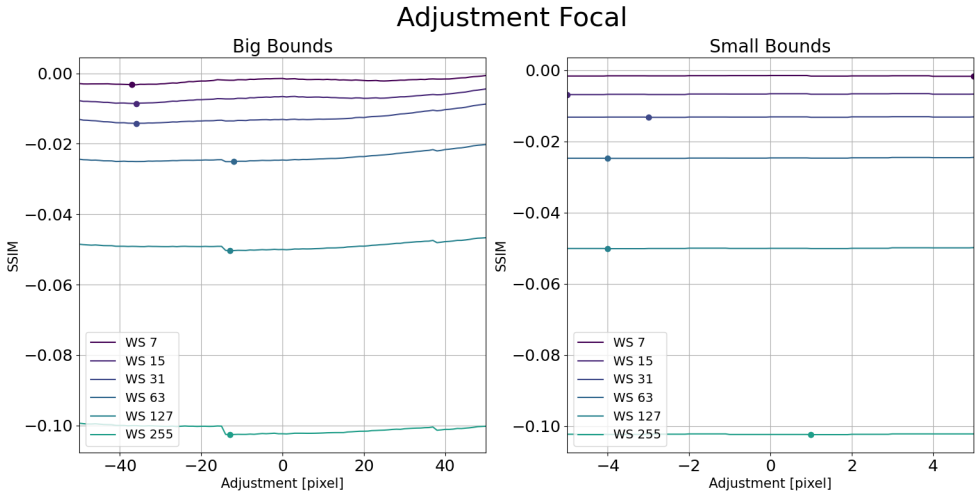

Figure 51: Handheld dataset SSIM Evaluation Focal length

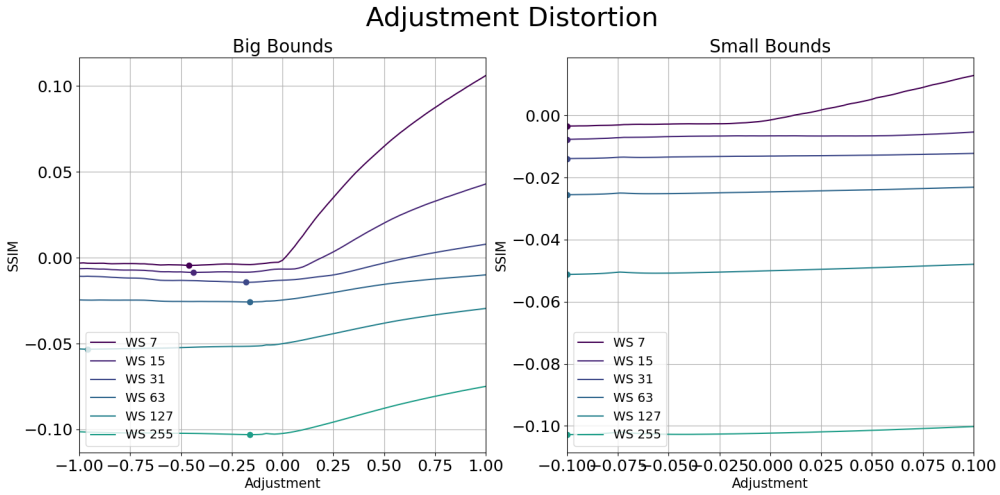

Figure 52: Handheld dataset SSIM Evaluation Distortion

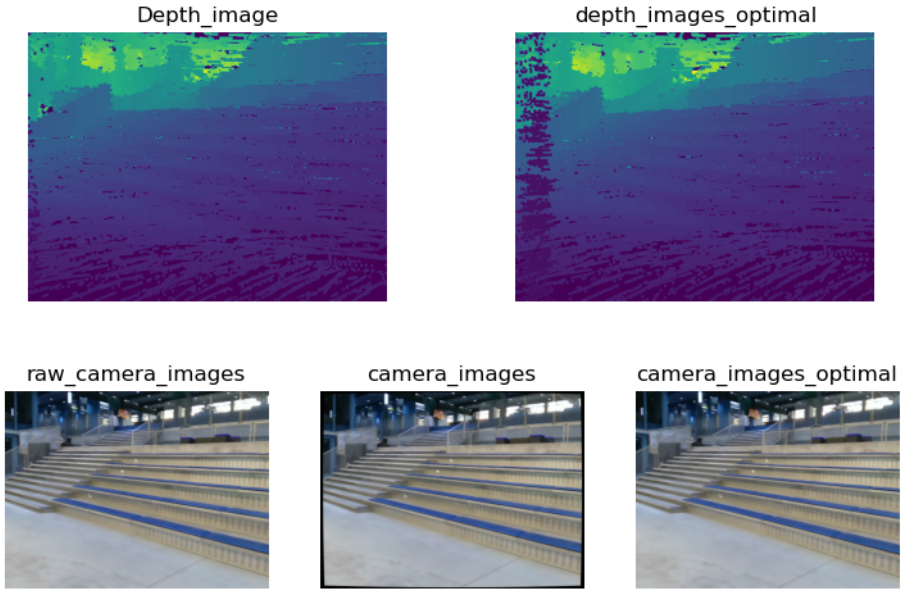

Figure 53: Handheld ground truth and optimal depth and camera images

Table 11: Error value for one degree of freedom SSIM-calibration for the Handheld combination dataset, the errors here is the absolute difference between peak value and ground truth value.

Window Size [pixel]	x [m]	y [m]	z [m]	Roll [°]	Pitch [°]	Yaw [°]	Focal [pixel]	Distortion
7	0.12	0.09	0.88	0.54	2.16	1.80	37.00	0.46
15	0.10	0.09	0.76	0.54	0.36	1.98	36.00	0.44
31	**0.03**	**0.06**	0.40	0.18	**0.18**	1.98	36.00	0.18
63	0.06	**0.06**	0.40	0.18	0.54	**1.62**	**12.00**	**0.16**
127	0.08	**0.06**	0.40	**0.00**	0.72	1.98	13.00	0.96
255	0.18	**0.06**	**0.16**	0.18	2.88	1.80	13.00	**0.16**