DEGREE PROJECT IN VEHICLE ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Automatic Online Calibration Between Lidar and Camera

YI YANG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Master Thesis

Automatic Online Calibration Between Lidar and Camera

YI YANG
yiya@kth.se

Supervisor: John Folkesson
Examiner: Patric Jensfelt
Host Company: Scania AB
Local Supervisor: Henrik Felixson
KTH Royal Institute of Technology
Date: October 31, 2019

Abstract

Online calibration between the Lidar and camera sensors is a prerequisite for the sensor fusion in autonomous driving. This master thesis project proposes a real-time algorithm to find six calibration parameters, (X, Y, Z) for translation, $(roll, pitch, yaw)$ for rotation between the coordinates of two sensors. The algorithm can achieve two goals. First, it can detect miss-calibration as an inspector of the system to ensure it keeps well-calibrated. Moreover, The algorithm also can correctly refine the parameters during a vehicle's movement if the initial parameters contain some error. Experiments are tested with the public available KITTI dataset. The results from the experiments are competitive or even better results for some parameters compared with other existing methods.

Sammanfattning

Online-kalibrering mellan Lidar och kamerasensorer är en förutsättning för sensorfusion i autonom körning. Detta examensarbete föreslår en realtidsalgoritm för att hitta sex kalibreringsparametrar, (X,Y,Z) för position och (roll, pitch, yaw) för rotation mellan de två sensorernas koordinatsystem. Algoritmen kan uppnå två mål. Först kan den upptäcka misskalibrering genom automatisk inspektion av systemet för att säkerställa att det håller sig välkalibrerat. Dessutom kan algoritmen förfina parametrarna under fordonsrörelsen och därmed rätta till eventuella fel i de initiala värdena. Experiment utfördes med publikt tillgänglig data från KITTI. Resultaten från experimenten är minst lika bra eller bättre för vissa parametrar jämfört med andra befintliga metoder.

Contents

Chapter 1

Introduction

1.1 Background

Autonomous driving is not a dream anymore, but it is coming. In 2018, the Society of Automotive Engineers (SAE) defined a six-level of driving automation from no automation to full automation shown in Figure 1.1 [1]. Currently, modern vehicles can reach Level 2. Drivers are assisted by some functions such as adaptive cruise control, lane keeping assistance, collision avoidance, automatic parking, etc.; however, they are still entirely responsible for the driving and need to be ready to control the vehicle anytime [2]. Many vehicle manufacturers are aiming to reach Level 3 or 4. From Level 3, drivers are not required to control the driving fully, instead, some functions are allowed to monitor the environment. However, researchers and the public have intense discussions about ethics and regulation of such functions. For example, it is hard to determine who should be responsible for the machine in case of a failure. This part will be investigated more in Section 1.5.

In the autonomous driving, there are four main steps [3].

Perception Just like a human, first, the vehicle needs to perceive the environment via different sensors, called perception. It includes but is not limited to computer vision and sensor fusion. Computer vision uses the images or videos from a camera to identify the lanes, vehicles, etc. A modern method of image processing involves Machine Learning techniques, which show very successful results for classification and regression. Instead of a camera, there are also other common sensors like Radar (Radio Detection and Ranging) and Lidar (Light Detection and Ranging) to supplement camera. This project belongs to the perception part, which aims to do the calibration between the camera and

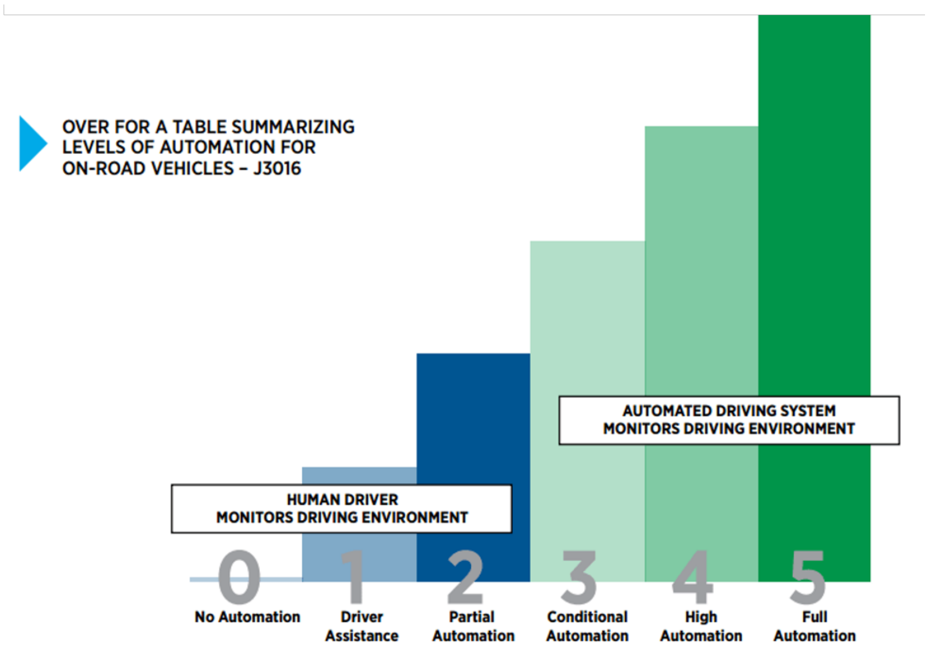


Figure 1.1: Five levels of autonomy [1]

Lidar sensors, as the foundation of these two sensors' fusion. More introductions and related works about calibration are shown in Chapter 2.

Localization and Mapping After gaining some information from the perception about the surroundings, the next question to be solved is to determine the position and direction of the autonomous vehicle with respect to the environment. It is called ego pose/motion estimation. In many cases, there is no prior knowledge of the environments, and the vehicle does not know where it is, then this case is called Simultaneous Localization and Mapping (SLAM). Currently, the state-of-the-art SLAM methods often use a stereo camera, or laser points, or both of them; meanwhile, they show excellent performance on the odometry estimation. Based on the KITTI Odometry Benchmark [4], the top 30 algorithms are quite accurate with less than 1% translation. Top 1 method *V-LOAM* [5], uses visual and Lidar sensor, even with only 0.56 % translation.

Path Planning With the map, after determining a destination point, the next step is to do the navigation. Motion planning is a long term research topic. For the vehicle, it does not only need to consider the fastest path, but also other vital aspects such as a dynamic map, collision avoidance, driving comfort. Notable

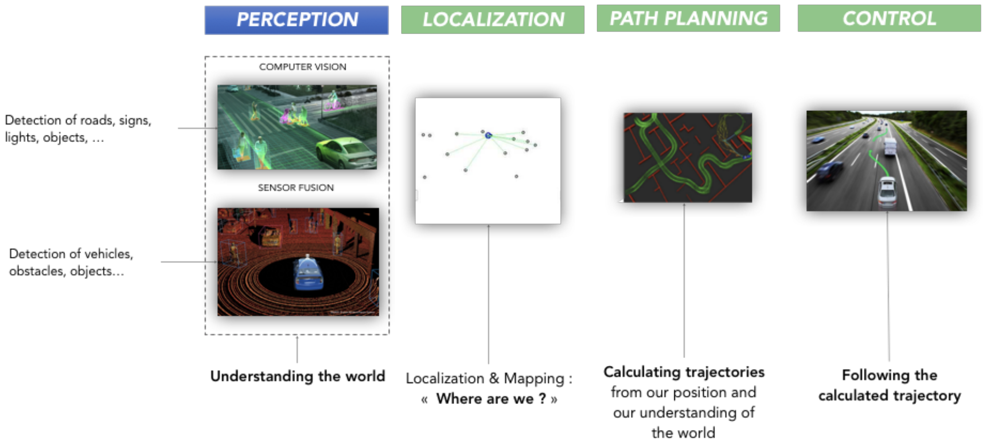


Figure 1.2: Four main operation steps of autonomous vehicle [3]

algorithms which are widely applied includes Dijkstra, A*, D*, Probabilistic roadmap (PRM), Rapidly-exploring random tree (RRT).

Control Given a planned reference motion state, finally, the last step is to control the actuators, including motors, steering wheel, and breaks.

Multi-sensors system is used for the perception and vision field for the vehicle to have a robust and detailed understanding of its environment. Colors and features are quickly captured by the camera under a good light condition. However, they are easily influenced by light intensity, such as the shadow of bright lights; moreover, cameras are not good at the range measurement compared with Radar and Lidar. The advantages of Radar and Lidar are accuracy and precision of the distance measurement. For Radar, it uses the Doppler effect to measure a speed; for Lidar, it uses an infrared sensor of laser pulses. In general, a Lidar has many layers (commonly 4, 8, 16, 32, 64 layers). It gets a point if one beam hits an object and then reflects. At a high frequency, a vast data of point cloud can be accumulated and collected. Take Velodyne HDL-64E as an example. It can generate up to 2.2 million points per second [6], which gives the vehicle an incredibly clear 3D image of the world. Recently, dense Lidar data is also used to do object detection with deep learning techniques.

1.2 Research Question

Lidar and camera are common sensors for the perception of autonomous or advanced driving. Both can compensate each other to adapt to different scenarios. Therefore, it is useful and important to fuse different data streams and integrate them into one robust multi-sensor system. However, before the sensor fusion, calibration needs to be done in order to transfer information between each data flow. This is the research goal for the thesis. The research questions can be specified as,

- how to calibrate a Lidar and a camera online efficiently,
- how robust the calibration algorithm is, and
- what performance of it compared with other different methods.

Sensors' calibration is the prerequisite for multi-sensor fusion system. The key process of the thesis project is the extrinsic calibration between Lidar and camera systems. The mathematical model will be introduced in Section 2.1.

1.3 Research Objective

The objective of the thesis is first to find the transformation between a Lidar and camera coordinates by calibration. Then visualize the fused result- the point cloud map with marked object detection from the camera on it. As for the expected scientific results, the refined calibration parameters are supposed to have a low error variant of translation and rotation parts with respect to the ground truth values provided by the dataset.

The innovation part is the online calibration part. Currently, more focus on sensor calibration is on offline methods using a chessboard-like pattern. It gets pretty accurate results but with less flexibility. Lidar and camera in an autonomous vehicle are calibrated by detecting a unique pattern from both sensors. The calculated calibrated values are rigidly fixed and cannot be adaptive. For instance, if there is a geometry change between the Lidar and camera during the moving, then the vehicle needs to stop and be calibrated again using the same pattern. However, the online calibration proposed here is an adaptive method which can continuously refine the previous calibration parameters and also inspect the sensors' system in case of sudden severe miss-calibration.

1.4 Outlines of Thesis

This paper presents the work of the master thesis in the following structure.

First, the mathematical model of Lidar-camera calibration and its related research solutions are introduced in Chapter 2. Two categories of the solutions-offline and online are introduced. Literature studies about these two methods and the comparison between them are fully researched. In Chapter 3, the methodology of the thesis project is presented. An overall pipeline of the whole architecture is introduced. Then data processing for the images and Lidar points is explicitly described with the optimization to find the updated calibrated value.

Next in Chapter 4, the results on the KITTI data set is presented. Moreover, the discussion is further presented.

Conclusions of the thesis project are drawn in Chapter 5.

1.5 Ethical, Sustainability, and Society

In recent years, with one of the most buzzworthy topics - autonomous driving, many issues about the relevant regulation, sustainability, ethics, and social impact raised by it have aroused heated public discussion. Questions come from many aspects, such as how to define an ethical behavior of autonomous vehicles when it needs to make a decision, how much the machine system including perception and navigation can be trusted, what negative influences it might bring, etc.

For instance, there is a famous scenario called *dilemma situation* [7]. Debates are generated about how autonomous vehicle should behave if there exists an unavoidable collision. According to a survey, taken among 2.3 million people all over the world, published in *Nature*, moral principles of people from different countries vary a lot [8]. Iyad Rahwan, a co-author of the survey from Massachusetts Institute of Technology, said that there did not exist "a perfect set of rules" which can be agreed with the public in all regions, due to the culture difference of varied participants in the survey. However, he also emphasized such moral decision happened daily for the human driver as well. Barbara Wege, from the German carmaker Audi, believed that actually, autonomous driving would increase the safety compared to a human driver, but since the accidents caused by autonomous driving always accompany with much more public attention, it gives the public an unsafe impression.

Chapter 2

Literature Study

2.1 Basic Equation

The calibration process of Lidar-camera system includes to find the *intrinsic* and *extrinsic* matrices for the camera and Lidar. As shown in Figure 2.1, the problem is mathematically defined as,

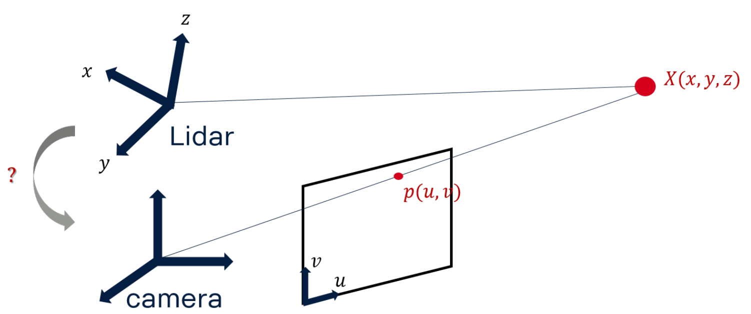


Figure 2.1: Projection from 3D Lidar point to 2D image pixel

- *how to find the perspective projection from 3D Lidar point* $X = [x, y, z, 1]^\top$ *to image pixel* $p = [u, v, 1]^\top$ *in homogeneous coordinates,*

$$p = K\underline{R[I|T]}X$$

which is explicitly

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where, $\boldsymbol{K}$ is the intrinsic camera parameter including the focal length (f_u, f_v), skew γ which is often close to zero, principal points (u_0, v_0). $\boldsymbol{R}[\boldsymbol{I}|\boldsymbol{T}]$ is the extrinsic matrix with the translation T and rotation R. It is a 6 degree of freedom matrix, with 3 from $T(x, y, z)$ and 3 from $R(\theta_{roll}, \theta_{pitch}, \theta_{yaw})$. Here the **focus** of the paper is on the **extrinsic** calibration.

First the 3D point X in Lidar coordinate is projected to 3D camera coordinate using $\boldsymbol{R}[\boldsymbol{I}|\boldsymbol{T}]$ with translation and rotation information, which is called extrinsic calibration. It is the aim of the thesis. After that, 3D camera point to the 2D image pixel p using intrinsic K. In this case, the intrinsic matrix is assumed to be already provided by the camera supplier.

2.2 Related Work

For the approaches for the extrinsic calibration of Lidar-camera system, there are two groups - offline and online.

For both of them, the goal is to find the rotation and translation matrices $\boldsymbol{R}[\boldsymbol{I}|\boldsymbol{T}]$. The idea of the solution is to find pairs of 3D points X and their corresponding 2D image pixels p. Therefore, with known K, an optimal $R[I|T]$ can be calculated with optimization method.

There are differences between them as well. Offline methods use special pattern to find accurate pairs of X and p. Hence it gets more accurate results. For the disadvantages, one of them is that it lacks of flexibility since it is only used with the pattern when vehicle is still. For online calibration, however, it works during vehicle's movement by find the pairs with natural scenes such as roads, buildings. It is not so accurate compared to offline calibration, but it can use to detect a sudden huge miss-calibration and give a warning while moving. It is critically needed as an inspection. Moreover, it allows to refine the calibration continuously with a limited range. Thus online method uses results of offline method as the initial values when the vehicle starts to move.

2.2.1 Offline Calibration

One of the most common methods for calibration is to use markers or checkerboard-like pattern. In 2004, Zhang and Pless [9] first proposed a method using planar checkerboard pattern observing from multiple views by a camera and 2D laser scan with the geometric constraints. Then the polygonal pattern calibration methods become to be an important branch as well as in 3D Lidar calibration. The consequential other methods further developed.

In early years, manual intervention like annotation was still needed in popular methods like [10] in 2007 and [11] in 2009. Now more methods are automatic [12–14] to decrease the error and speed up the process. Geiger *et al.* [12] proposed a way to use a single shot for calibration with multiple checkerboards. For camera-to-camera calibration, it is done by matching checkerboards between images which is extracted from corner detection. Then for camera-to-range calibration, there are three steps. First, to find the plane from point cloud data; second, initial the transformation by randomly selecting plane from images and range data; finally, make the refinement loop by minimizing the point-to-point distance and non-maximum suppression. The method is used for the KITTI official calibration. Beside checkerboards, there are methods using different patterns like triangle [15,16], diamond-shape [16], v-shape [17], circles [18], ordinary box [19].

There is one challenge for the checkerboard-like method - the reflectivity difference between black and white for Lidar. An example of the "systematic range-reflectivity-bias" is shown in Figure 2.2. To solve it, in [16], Park *et al.* recommended to use bright monochromatic color like white for the board, which meanwhile should be distinctive to the background color. He applies a novel method to find vertices from Lidar using the adjacent sidelines on a board of known length. The method can be applied for a robust calibration even for low-resolution Lidar which has 16 or 32 beams only. Finally, only three corresponding pairs which represent three triangle vertices from camera and Lidar, are used to calculate the transformation. Singular value decomposition (SVD) is used to update the transformation, and Levenberg-Marquardt algorithm as an optimization to get the final result.

Especially for the calibration between the stereo camera and Lidar, Guindel *et al.* [14], proposed a smart pattern which makes it easy to calibration even for low-resolution Lidar. Figure 2.3 shows the calibration target pattern. It is a plane with four symmetric circular holes. The idea is to estimate four pairs of very accurate circle center from both stereo and Lidar coordinates to find out the extrinsic calibration parameters. Figure 2.4 shows the pipelines of the extraction of four center points from both Lidar and stereo. Even the stereo, it uses the depth estimation to transfer images into the point cloud, which is much denser than the Lidar points. Then the method of Semi-Global Matching (SGM) is applied for stereo to have accurate depth estimation. After coarse target segmentation and then fine circle segmentation, the set of transformation parameters can be calculated out by minimizing the point-to-point distances via Iterative Closest Points (ICP) algorithm.

Instead of using a polygonal plane, there are other methods [13]. A novel

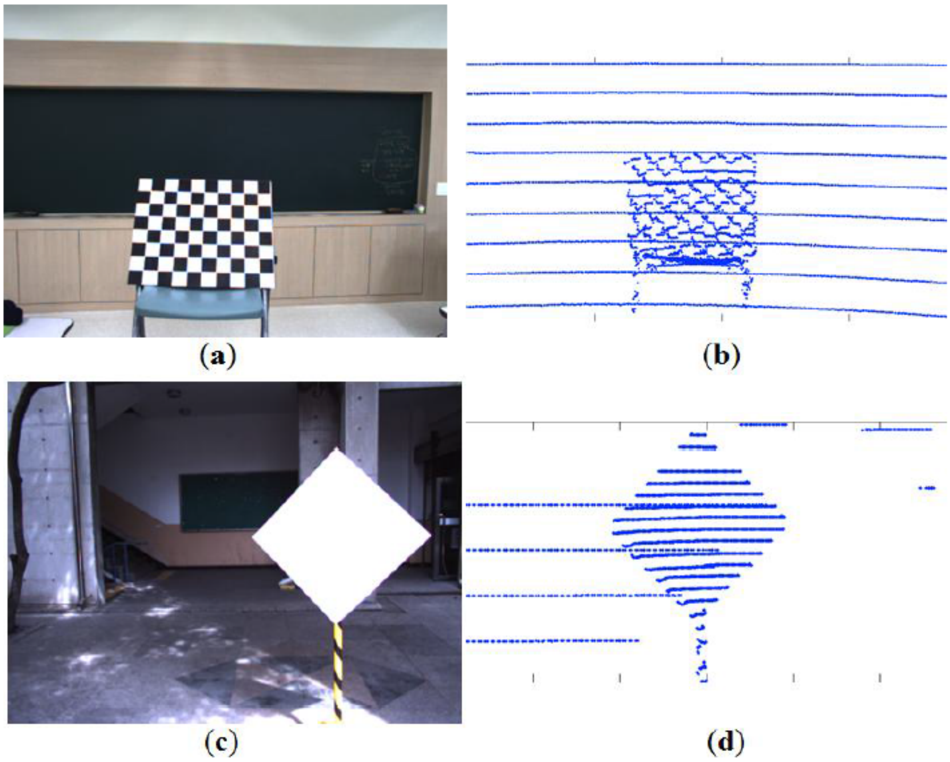


Figure 2.2: Scanning on the checkerboard (top) v.s. monochromatic board (bottom) with Velodyne HDL-32E [16]

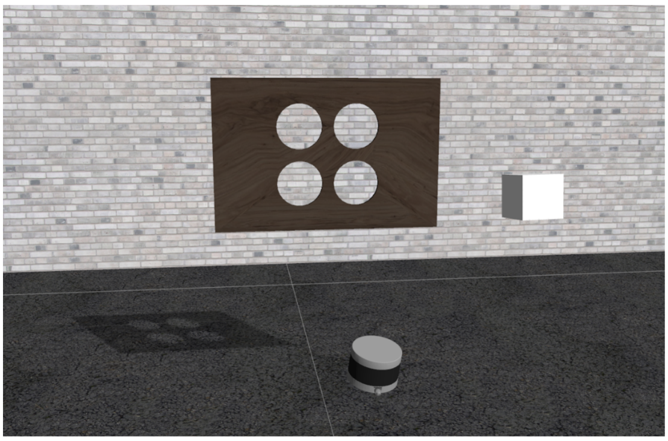


Figure 2.3: Calibration pattern proposed by Guindel *et al.* [14]

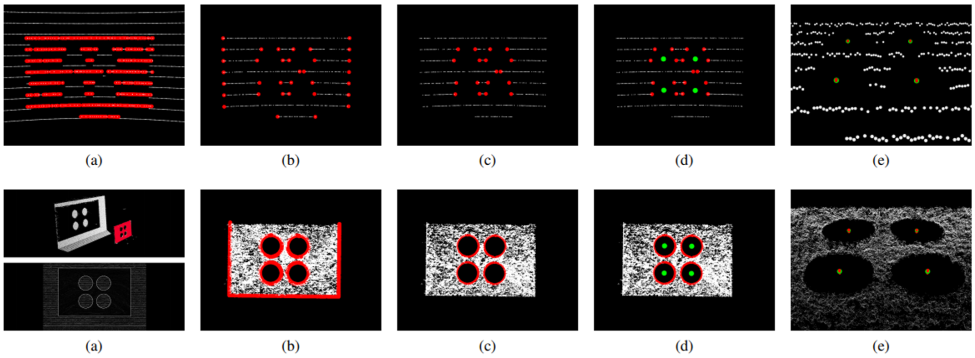


Figure 2.4: Pipeline of extraction of four center points from Lidar (top) / stereo (bottom) [14]

approach for entire multi-system including inertial measurement unit (IMU), cameras and Lidars are proposed by Taylor *et al.* [13]. They used the motion of the system to compare with the motion calculated by only IMU or cameras or Lidar, to get the extrinsic parameters. Then they refined them using the overlapping view from them.

2.2.2 Online Calibration

Recently online calibration has attracted many researchers' attention [20–23]. The advantage of it is that the process can happen while moving. It does not need a special calibration marker or board to help. If there is a position change for the sensors while moving, it can quickly refine the transformation, which is unavailable for offline calibration. In general, online calibration starts with a correct initialization from an offline process and updates the transformation online based on it.

Moghadam *et al* [20] use natural scenes and apply a line-to-line calibration which is more suitable for indoor scenarios. Napier *et al.* [22] used Lidar reflectance imagery for calibration. Their algorithm does not require an overlapping area from a camera and Lidar at the same time frame. It only needs that two sensors observe the same objects which can be at different times during the movement. The idea is to fuse multiple scenes at multiple time frames into one image for both Lidar and camera using the known vehicle's trajectory. Then do the optimization for calibration parameters to match the fused laser reflectance imagery with the fused image of the camera.

Instead of using reflectance information of all Lidar points, Levinson and Thrun [21] introduced two real-time algorithms using depth information with selected edge Lidar points. One can detect sudden miscalibration. Another optimizes the transformation continuously while moving, which is locally convex and near global optimum. They detected the edges from both camera and Lidar. For the camera, the calibration uses the inverse depth transform and edge detection. For Lidar, all depth-discontinuous points with weights are selected from each sweep data. They assumed that depth discontinuity of laser would be more matched onto image edges with accurate calibration, which is also used in [24] and verified of the strong robustness in [25].

Based on [24], more recently, there are further researched methods [23,26]. Blaga and Nedevschi [23] designed a similar calibration system. The novelty is adding the application of motion correction for distorted Lidar data via VICP algorithms [27] in the data preprocessing before the edge detection to further improve the accuracy. Moravec and Sara [26] found out it was vital to estimate the robustness of the transformation between Lidar and image edges. Each time when there is a new calculated transformation result, it needs to reach some thresholds to guarantee the robustness so that the result can be used and updated to the vehicle. They proposed a low-overhead robust model to evaluate stability. A delayed learning mechanism with AdaGrad stochastic gradient descent (SGD) is used for the approximation of optimization for the proposed cost function. Since the function is not converged in the beginning, in order to have the final result near the pre-calibration, they found the delaying learning process was critical.

Instead of looking for edge detection first, Pandey *et al* [28] proposed an algorithm which uses all overlapping data. It is under the assumption that given the accurate calibration, the correlation between the greyscale intensity of images and the surface reflectivity of Lidar should be a maximum optimum due to the texture consistency of the same observed objects. The cost function is written as a format of entropy which merges both, to indicate how much correlation between the information of objects from the camera and the Lidar.

Chapter 3

Methods & Implementation

In my thesis, the proposed method has a similar structure to the automatic online calibration method suggested in [21] but with improvement and comparison among different methods. The flow diagram is shown in Figure 3.1.

First, a proper initialization is calculated by some offline method, such as Geiger calibration web toolbox [12], or [14]. Then do the edge detection for both Lidar and camera data, respectively, to find the 3D-2D correspondences. After that, a new calibration matrix is calculated by maximizing a cost function with optimization. Finally, determine whether the new matrix is needed to be updated and replace the previous one by checking several conditions. The method is under an assumption that the edges (range discontinuity) from the Lidar point cloud should be projected into the image edges more frequently when the calibration is more accurate, which is also used in [21] and verified for the robustness in [25].

3.1 Image processing

For the images, the method aims to find out the discontinuities - edges. Each pixel is assigned a value to indicate how probable it is that this is an edge.

For each frame image $\boldsymbol{I}^t$ taken at time t, out of total n frame images $\boldsymbol{I}^{1:n}$, first the transform to grayscale is processed.

Then canny edge detection is applied to generate an output of the edge image $\boldsymbol{Edge}^t$. For the process of canny edge detection [29], there are five steps. Firstly Gaussian blur of kernel size 3 is used to filter out noises, then secondly intensity gradient strength and direction of each pixel are calculated. Thirdly, non-maximum suppression is used to remove outliners for edge thin-

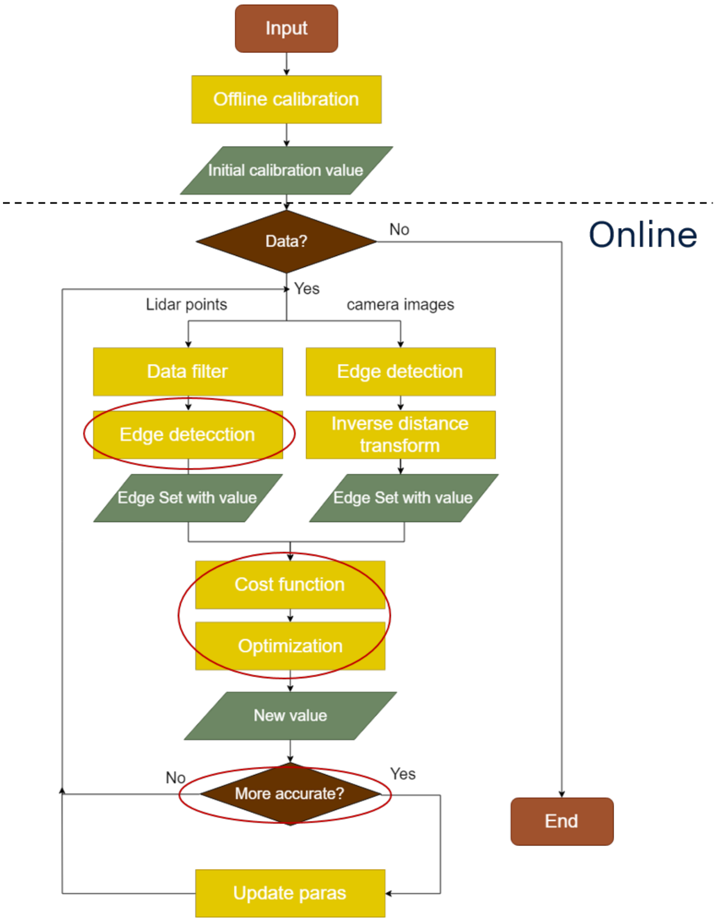


Figure 3.1: Flow diagram

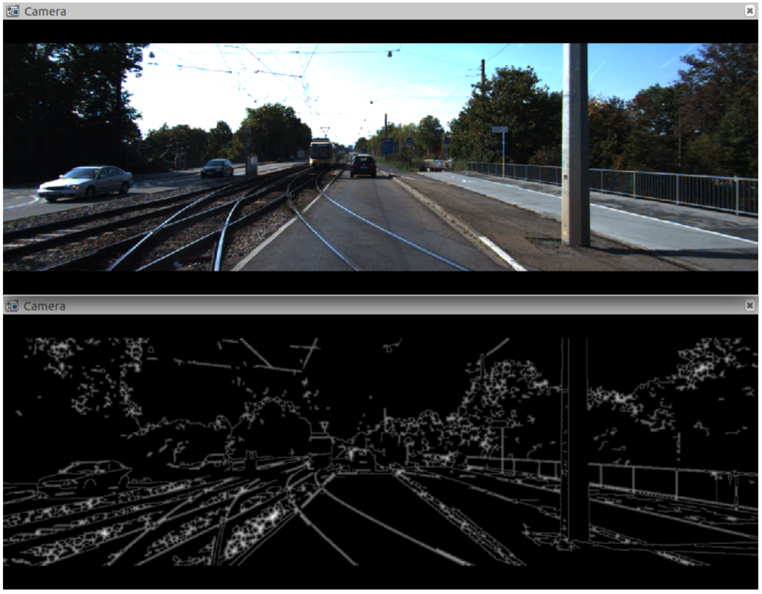


Figure 3.2: Raw image (top), distance transform image (bottom)

ning. Fourthly double thresholds are applied to remove edge noises further and classify strong and weak edges. The last step is to determine whether weak edge pixels are kept or not according to blob analysis such as dilation via checking its neighbors.

Next dilation [30] is added to make each value to be the maximum value among its 8 neighbors with 3×3 kernel size.

$$I^{u,v} = \max_{x,y \in [-1,1]:(x,y) \neq 0} Edge(u+x, v+y)$$

Finally, the inverse distance transform is used so that the value of each pixel $I_{u,v}$ of current image frame I gain a value which presents the distance to the closest edge pixels. That value indicates how possible it is that this is an edge. L2 distance type is chosen for accuracy. Normalization is also used to have the pixel value in the range from 0 to 1. Figure 3.2 shows the result of the image processing.

3.2 Lidar points processing

For Lidar point clouds, there are several steps.

Preprocessing First given the initial calibrated value from offline calibration and intrinsic camera parameters from supplier, Lidar points are projected into

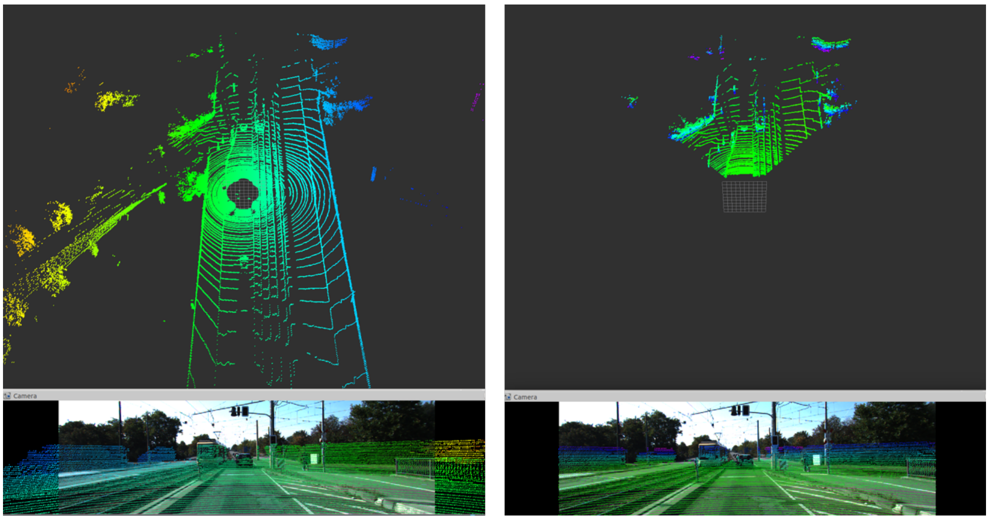


Figure 3.3: Raw data (left), filtered data (right). Point cloud (top), point cloud projected on the image (bottom)

the camera coordinate of the current image frame. Only the points projected on the same objects which can also be found in the image are kept, and others are discarded. The result of this step using KITTI 360°Lidar is shown in Figure 3.3.

Edge extraction based on [21] The edge points are calculated by the huge range difference between neighbors. For each point P^i of a beam, the neighbors are its left P^{i-1} and right P^{i+1} points from the same beam. Referred to the paper [21], the range gap value X^i of the point P^i is

$$X^i(P^i) = max(R^{i-1} - R^i, R^{i+1} - R^i, 0) \tag{3.1}$$

where R is the range of corresponding point P. Note that only the point whose range is much smaller than those of neighbors can be considered as an edge point. If the range is much larger than neighbors' ranges, it is excluded since the far point is reflected from the background.

The threshold for KITTI data set is chosen to set as 1 meter. It means that only the points whose gap value X^i is larger than 1 meter are extracted out as an edge point to proceed further.

Further improvement of edge extraction In practice, if only applying Equation 3.1 introduced in the paper [21] for Lidar edge points detection, however,

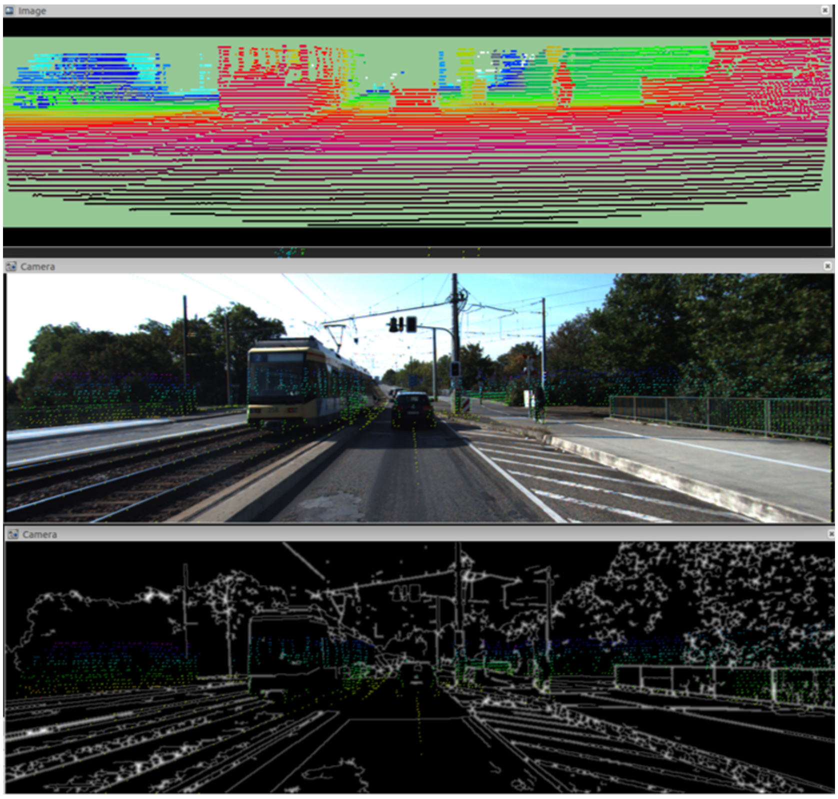


Figure 3.4: Range image (top), raw image with raw point could (middle), edge image with edge point cloud after Equation 3.1 (bottom)

might result in many undesired edge points such as discrete tree's points shown in Figure 3.4.

To further improve the result, a median filter is applied in the beginning to remove random noises but keep the sharp features. The method is found especially useful during data processing of Lidar point clouds due to the data discretization of Lidar. Moreover, small errors are Lidar also introduced during data collection. Since the sensor is spinning, actually the data are not collected simultaneously which also causes .

Meanwhile, more constraints are added to increase the probability of the edge points. Among the points collected from a same near-horizontal beam, for an edge point R_i (on an object), if the range gap exists between it and its leftmost point R_{i-1}, it means the left point belongs to background. R_i also requires a gap between it and other several left points $(R_{i-2}, R_{i-3}, R_{i-4})$, which all belong to background . Meanwhile, its rightmost point R_{i+1} is supposed to belong to the same object. Thus it requires a small range difference between the point R_i and its rightmost point.

$$
\begin{aligned}
G_{l1} &= R_{i-2} - R > T_{high} \\
G_{l2} &= R_{i-3} - R > T_{high} \\
G_{l3} &= R_{i-4} - R > T_{high} \\
G_{r1} &= abs(R_{i+1} - R) < T_{low},
\end{aligned}
$$

where T_{high} defines a range gap, T_{low} defines a small range difference on the same object. In the case of KITTI data set, T_{high} is set as 1 meter, T_{low} is set as 0.5 meter.

To cluster the discrete edge points and further remove outlines, RANSAC line model is used. Top 12 biggest line clusters remain according the the result of experiments. The effect is shown in Figure 3.5. Lines are clearly extracted out such as poles, trunks, rail tracks, a profile of the front vehicle, and remove outlines like tree's points. On the other hand, however, it also removes some small clusters of edge point and reduce data.

3.3 Cost function

After the edge detection from both Lidar and image data, then a cost function of calibration matrix is defined. The cost value indicates how much correlation between these two data sets. Since both camera and Lidar are observed same scene, they are supposed to have a strong correlation. More accurate the

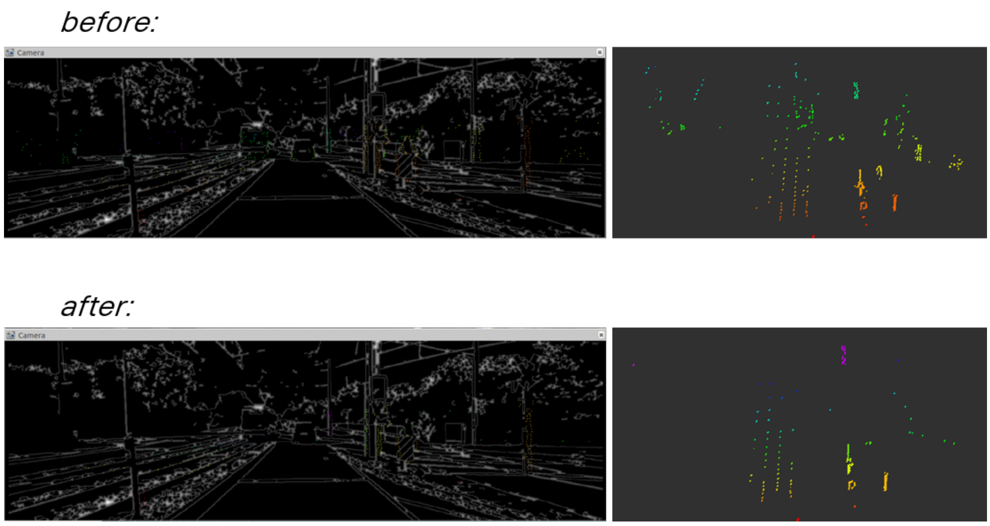


Figure 3.5: Lidar edge points before (top) / after (bottom) RANSAC line model. Lidar points project on the image edge (left), only Lidar points (right).

calibration matrix is, larger the value will be. To maximize the cost, given the current calibration parameter $\boldsymbol{\theta}$, in frames of window size n, for each frame $\mathcal{F}$, find all the Lidar edge points $X_{1:m}^{\mathcal{F}}$, for each point $X_i^{\mathcal{F}}$, and corresponding projected camera pixel $I_{u,v}^{\mathcal{F}}$ calibration parameters. Then the cost can be written as

$$Cost(\theta) = \sum_{\mathcal{F}=1}^{n} \sum_{i=1}^{m} \sqrt{X_i^{\mathcal{F}} \times I_{u,v}^{\mathcal{F}}} \tag{3.2}$$

Bilinear interpolation is used for calculating value of $I_{u,v}$ during projection from Lidar point to image pixel in order to make the gradient descent function more smooth. A global search for the optimal θ is not possible since the function is not convex. However, local search given a relatively high-accuracy initial θ is feasible even though there might be multiple optima.

Another constraint is added to verify the correctness of the calibration parameters using a grid search. Given a small perturbation (plus or minus a resolution) for each of the 6 parameters, translation $[x, y, z]$ and rotation $[\theta_{roll}, \theta_{pitch}, \theta_{yaw}]$, there will be $3^6 = 729$ values of cost in total. The 729 values include values of the original one and others of 728 cases shown in the Figure 3.6. For the translation, the perturbation is $0.01\ meter$; for the rotation, the value is $1°$. Then the percentage of cases with worse cost than the original one, P_C, is counted as,

$$P_C = \frac{\sum_{i=1}^{728} count + 1\ (if\ cost_i < cost_{original})}{728} \tag{3.3}$$

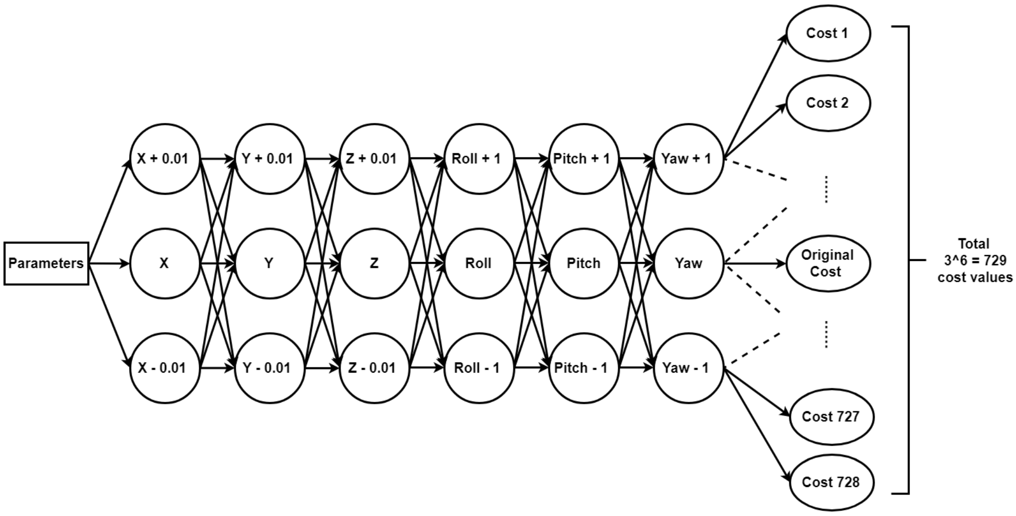


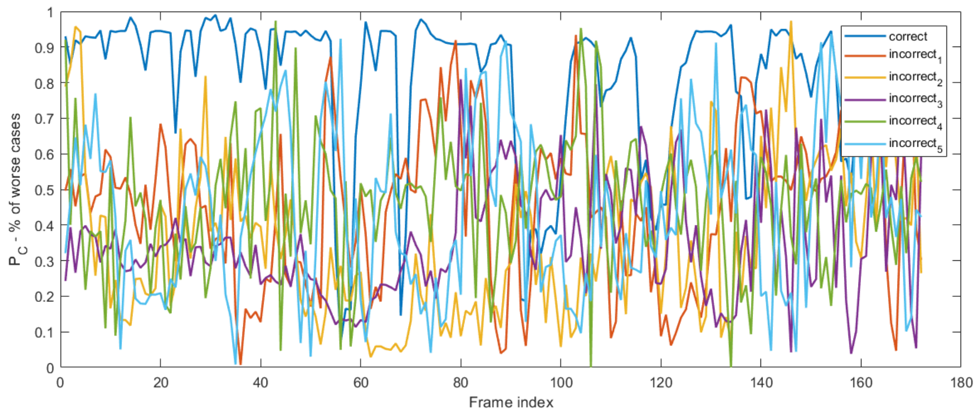
Figure 3.6: Calculate the worse percentage from 729 cost values

Unit (meter)	X	Y	Z
$Correct$	0.059	-0.075	-0.272
$Incorrect_1$	1.000	1.000	1.000
$Incorrect_2$	0.550	0.430	0.270
$Incorrect_3$	-0.450	-0.570	-0.730
$Incorrect_4$	-1.441	-1.575	-1.772
$Incorrect_5$	1.559	1.425	1.228

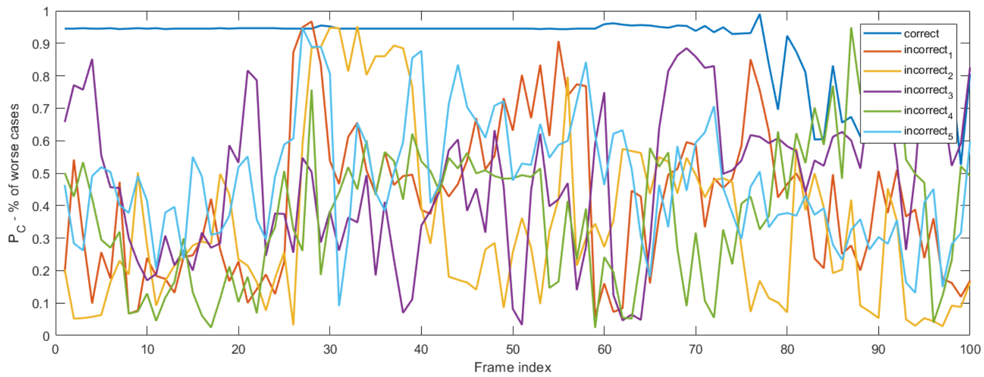
Table 3.1: Configuration of translation variables (X, Y, Z) used in 6 groups of tests

P_C is used to evaluate the correctness of the calibrations. In the experiment of the KITTI dataset, it is found out that there exists an obvious difference of P_C between correct and incorrect calibration values shown in Figure 3.7a. Figure 3.7a shows a comparison with the correct calibration parameters and five different incorrect values. The set of correct calibration parameter is provided by the KITTI offline calibration result, which is assumed to be regarded as the ground truth. Other five sets of incorrect parameters are randomly selected, whose translation values (X, Y, Z) have some variances with KITTI one. The variances range from 0.5 m to 1.5 m in three translation values (X, Y, Z). Table 3.1 shows the values used for Figure 3.7a.

In both Figure 3.7a and Figure 3.7b, for the correct calibration matrix, most of the time, the percentage of P_C is larger than 0.9 which is not shown for an incorrect set of values. P_C is one criterion to check the accuracy of the

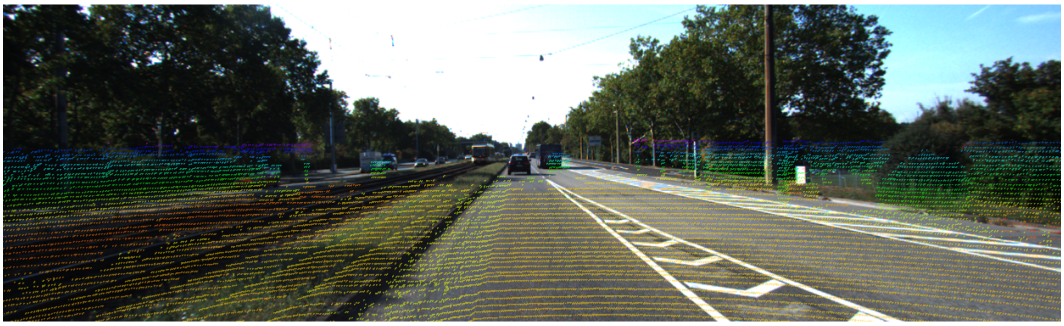


(a) window size of 10

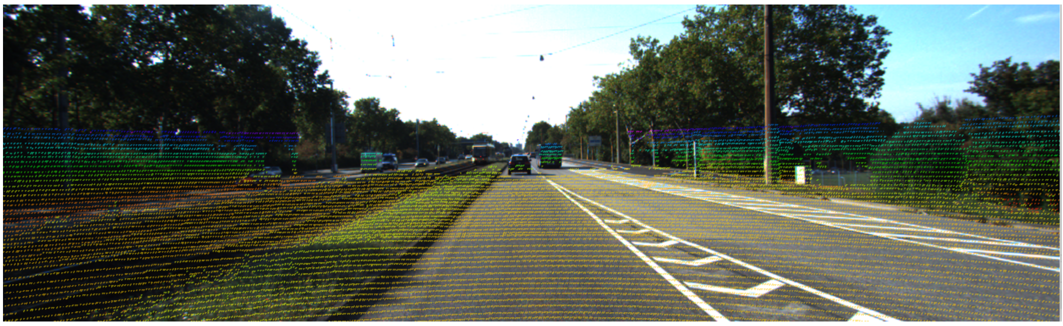


(b) window size of 20

Figure 3.7: Percentage of the worse cases of correct and wrong calibration parameters with small perturbation under different window size



(a) Miss-calibration



(b) Correct calibration parameters

Figure 3.8: Projection of the Lidar points on the images

calibration parameters. Figure 3.7a shows the result under the window size of 10, i.e. 10 continuous frames of the data. Figure 3.7b shows the result under the window size of 20. With increasing the window size, the curves become more smooth, and the distinction between the correct and incorrect translation matrix becomes larger.

3.4 Miss-calibration

Miss-calibration detection is a non-trivial function for autonomous driving. It works as an inspector, which checks whether sensors are always well calibrated. Once there exist uncalibrated sensors, or the position of sensors is suddenly changed, for example, due to accidental collisions, then it is imperative that the system knows it and the result of sensor fusion part under the miss-calibration should not be regarded to be convincing. For instance, take one frame from KITTI dataset, Figure 3.8a shows the wrong projection of the Lidar points on the images compared to the correct one in Figure 3.8b. This miss-calibration should be detected instantly.

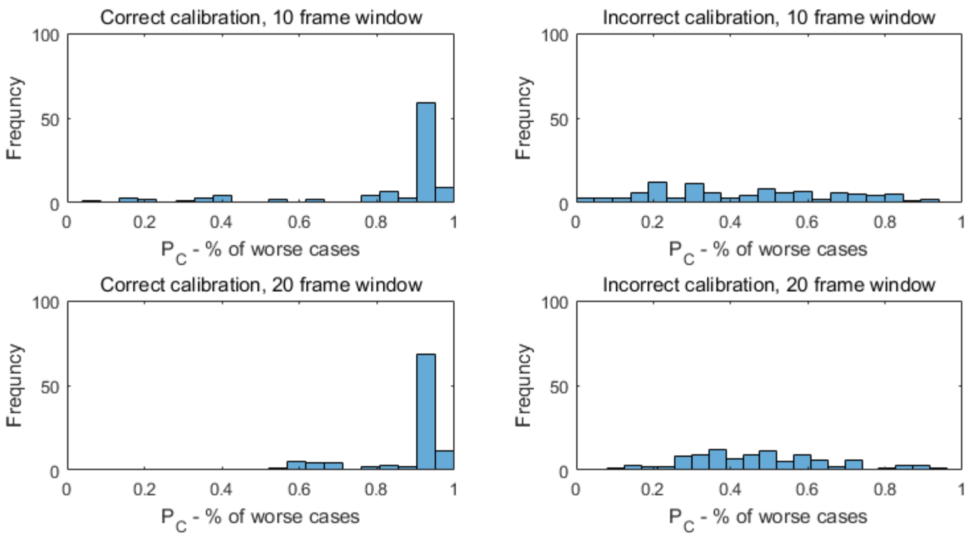


Figure 3.9: Histogram of the calibrated v.s. uncalibrated sensor. *Note that Frequency in y axis means the number of windows out of total 100 windows.*

In last section, it is mentioned that for the correct calibration matrix, most of the time, the percentage of P_C is larger than 0.9. However, how to quantitatively define the most of the time is the next step. Given a set of calibration parameter, to estimate whether it is miss-calibrated, a mathematical model of Gaussian distribution is used.

Using two groups of data in Figure 3.7a and Figure 3.7b (one is with correct calibration, another is with one of incorrect calibrations), four histogram graphs are generated with two different window sizes shown in Figure 3.9. From the histograms, there is an obvious distinction between correct and incorrect calibrations. Here a standard Gaussian distribution is used to fit into the histogram distribution to gain the optimal mean μ and standard deviation σ. These two parameters vary a lot in the situations of correct or incorrect calibration. Therefore, they can be used to check the miss-calibration.

$$D = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.4}$$

As shown in Figure 3.9, for the correct calibration, the μ of 10 frames and 20 frames are 0.8166 and 0.8945 , σ are 0.2355 and 0.1114, respectively. For the incorrect calibration, μ are 0.4369 and 0.4772, σ are 0.2369 and 0.1851. The difference of μ is huge. The complete results of μ and σ of the fitted Gaussian distribution of six groups are shown in Table 3.2. According to it,

	window size of 10		window size of 20	
Unit (%)	μ	σ	μ	σ
$Correct$	0.816571 [0.769846, 0.863295]	0.235481	0.894554 [0.872453, 0.916655]	0.111384
$Incorrect_1$	0.436859 [0.389851, 0.483867]	0.236909	0.477229 [0.440506, 0.513952]	0.185076
$Incorrect_2$	0.457833 [0.42118, 0.494486]	0.184722	0.389218 [0.344561, 0.433875]	0.225062
$Incorrect_3$	0.332455 [0.305903, 0.359008]	0.133819	0.472401 [0.427969, 0.516832]	0.223925
$Incorrect_4$	0.296639 [0.256831, 0.336448]	0.200626	0.346049 [0.29499, 0.397109]	0.257327
$Incorrect_5$	0.453580 [0.41167, 0.49549]	0.211218	0.420206 [0.374863, 0.465549]	0.228519

Table 3.2: Gaussian distribution fitted parameters: μ & σ of 6 groups of calibration tests (in Table 3.1) with 10 / 20 window size. *Note: the interval next to estimated μ means the 95% confidence intervals.*

the quite narrow 95% confidence interval for μ shows the accuracy of estimated μ. In both Table 3.2 and Figure 3.9, for the correct calibration parameters, μ is above **0.8**. In incorrect calibrations, however, μ is below **0.55**. With window size increasing, the value is more validated with more data and reduced random errors. Hence μ can be used to quantitatively determine whether a set of calibration parameters is miss-calibrated or not.

3.5 Optimization

The gradient descent method proposed by Barzilar & Borwein [31, 32], is applied to optimizing the cost function iteratively. Equation 3.5 shows how the parameter θ updates,

$$\theta_{k+1} = \theta_k + \eta_k \frac{G_k}{||G_k||}, \tag{3.5}$$

$$where\ G_k = \nabla_\theta J(\theta; x^{(i)}) \tag{3.6}$$

where $\nabla_\theta J$ is the gradient of θ with respect to each data $x^{(i)}$, η is the step size (or called learning rate). It required a differentiable objective function, As mentioned before, in this case, the cost function can be assumed to be differentiable within a small local range. There doesn't exist a math formula for the derivative of the cost function with calibration parameters θ (X, Y, Z, roll, pitch, yaw) since it is impossible to write the image values and point cloud range directly into a cost function. Thus, it is approximately estimated by the numerical gradient.

η is an adaptive step size. It is calculated as shown in Equation

$$\eta = \frac{s_k^\top s_k}{s_k^\top g_k} \tag{3.7}$$

where

$$s_k = \theta_k - \theta_{k-1}, g_k = G_k - G_{k-1} \tag{3.8}$$

, $||\cdot||$ represents the Euclidean norm. Algorithm 1 shows more details.

Algorithm 1 Gradient descent

Input: Numerical approximate gradient: resolution for translation δ_t = 0.01m, for rotation δ_r = 0.1°. Inital upper and lower threshold for the saturation of step size η: for translation, $[\eta_t^u, \eta_t^l] = [0.05, 0.0005]$, for rotation, $[\eta_r^u, \eta_r^l] = [0.05, 0.0005]$. Max step=300.

Output: Updated parameters $\theta_{calibration}(X, Y, Z, roll, pitch, yaw)$.

while iteration < Max step **do**

  Estimate gradient $G_k : (G_X, G_Y, G_Z, G_{roll}, G_{pitch}, G_{yaw})$.

  Get $s_{k,t}$, $s_{k,r}$, $g_{k,t}$, $g_{k,r}$ $\leftarrow$ Equation 3.8 for both translation (X, Y, Z) and rotation $(roll, pitch, yaw)$.

  Calculate $\eta_t, \eta_r \leftarrow$ Equation 3.7, and do the saturation filter using thresholds of $\eta_t^u, \eta_t^l, \eta_r^u, \eta_r^l$.

  Calculate the Euclidean norm $||G_t|| \leftarrow (G_X, G_Y, G_Z)$, and $||G_r|| \leftarrow (G_{roll}, G_{pitch}, G_{yaw})$.

  Gain estimated $\theta_{k+1,i\in(X,Y,Z)} \leftarrow \theta_{k,i\in(X,Y,Z)} + \eta_t \frac{G_{k,i\in(X,Y,Z)}}{||G_t||}$. $\theta_{k+1,i\in(roll,pitch,yaw)} \leftarrow \theta_{k,i\in(roll,pitch,yaw)} + \eta_r \frac{G_{k,i\in(X,Y,Z)}}{||G_r||}$ (Equation 3.5).

  if cost(θ_{k+1}) > cost(θ_k) **then**

    Update parameters θ_{k+1}.

  else

    Decrease threshold $\eta_t^u, \eta_t^l, \eta_r^u, \eta_r^l$ and resolution δ_t, δ_r for numerical gradient.

    if $delta_t$ < 0.001 (meter) **then**

      Break while loop.

    end if

  end if

  iteration $\leftarrow$ iteration + 1

end while

Chapter 4

Results & Discussion

4.1 Data

The implementation is based on the Robot Operating System (ROS) developed in C++, using the library, including Point Cloud Library (PCL), OpenCV, Eigen.

The data used in the master thesis is KITTI dataset [33]. It includes a Velodyne HDL-64E Lidar with *10 Hz* frequency, *2 cm* distance accuracy, *120 m* range. The field of view is *360* ° horizontally with *0.08* ° azimuth and *26.8* ° vertical direction. For the camera, the experiment uses the left PointGray Flea2 color camera to calibrated with Lidar.

4.2 Miss-calibration

As presented in Section 3.4, detection of miss-calibration is implemented.

An experiment is done to see how the P_C, the percentage of worse cases, changes when the initial correct parameters (see Table 3.1) are suddenly shifted by different centimeters D_{cm} (from $-10\ cm$ to $10\ cm$) for all three (X, Y, Z) translation parameters at the time $t = 3.8285\ s$. The upper figure in Figure 4.1 shows the result when D_{cm} is positive, while lower figure shows when D_{cm} is negative. First, the algorithm responds quickly in less than **0.0005s**, which is concluded from that the latest response among all the cases of different D_{cm} shows at the time $t = 3.8290\ s$. Second, P_C shows a obvious decrease below the value of 0.8, when the positive shift is larger than $2\ cm$, or negative shift is less than $10\ cm$. The interval where P_C is stably larger than 0.8 is about $12\ cm$; therefore, for any translation shift which is larger than $12\ cm$ from the correct calibration values, the miss-calibration algorithm can detect it out.

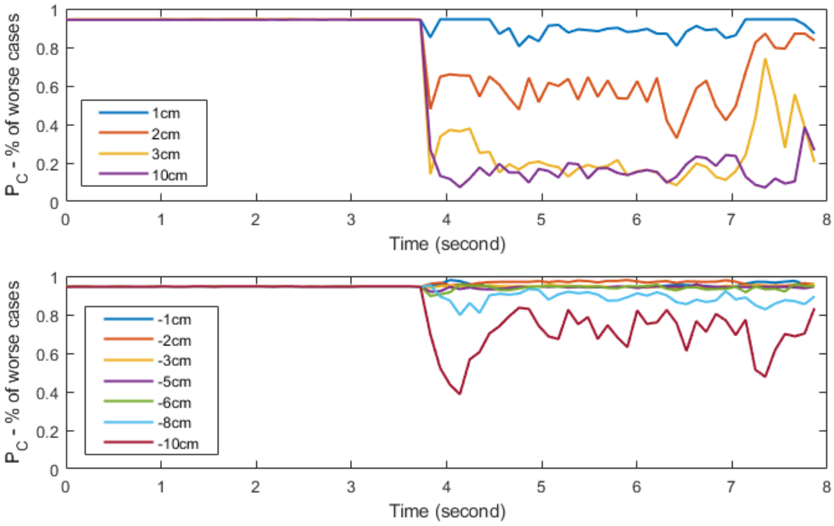
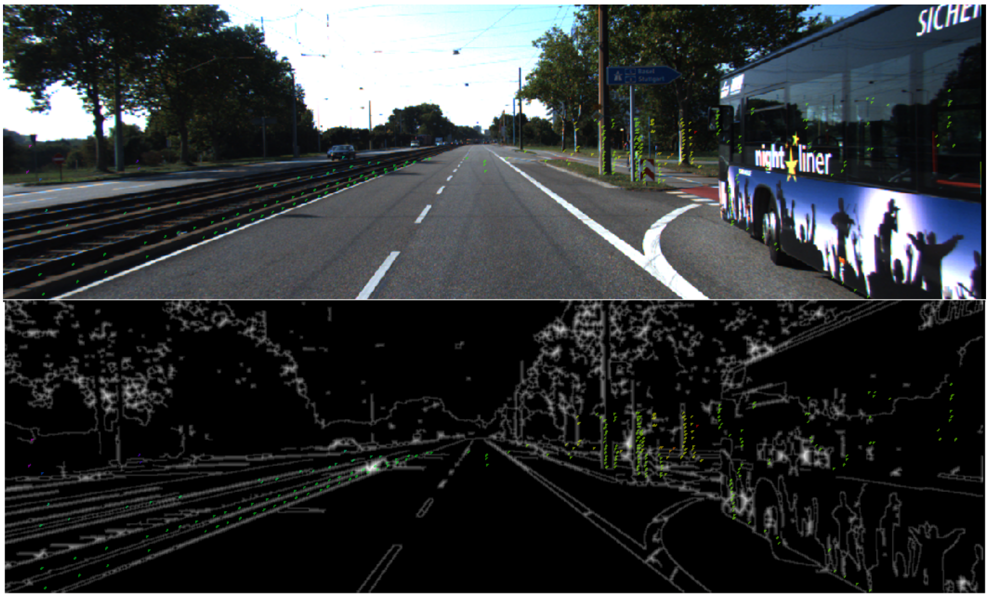


Figure 4.1: Miss-calibration detection: how P_C changes when the correct parameters are suddenly shifted by different centimeters D_{cm} at the time $t = 3.8285\ s$. Upper: positive shift D_{cm}, lower: negative shift D_{cm}. *(with 10 fps)*

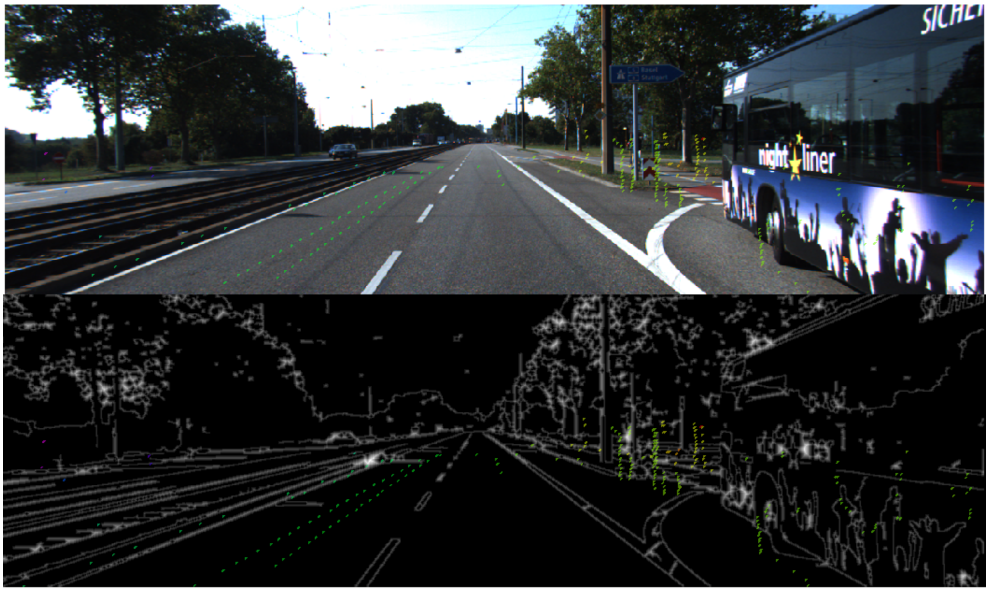
Figure 4.2a and 4.2b show two examples where the algorithm can detect the miss-calibration with $10\ cm$ and $100\ cm$ translation shift respectively. Even for the case of $10\ cm$, the shift error is not obvious to be observed from image; however, the miss-calibration algorithm can detect it.

Note that it is reasonable that the threshold ($2\ cm$) of positive shift is different from the threshold of negative shift ($10\ cm$) since there is no property of symmetry for the environment (input data).

A similar experiment is also done for the rotation angle shift $(roll, pitch, yaw)$ from $-2°$ to $2°$ shown in Figure 4.3. Note that the shift is added on all three parameters together (see in Table 4.1), same as translation shift added on all (X, Y, Z). As the figure indicates, the positive shift angle threshold is smaller $0.1250°$, and negative angle threshold is smaller than $0.5000°$. Beyond the thresholds, P_C shows an obvious decreasing below the value of 0.8. The interval of *quite accurate calibration* is around $0.6250°$ $(= 0.1250° - (-0.5000°))$. Hence, for the angle shift on the rotation larger than $0.6250°$ from the correct calibration values, the miss-calibration algorithm can detect it.



(a) Miss-calibration for 10 cm translation shift



(b) Miss-calibration for 100 cm translation shift

Figure 4.2: Miss-calibration of incorrect projection from Lidar points to raw/edge image. Both can be detected out from the algorithm.

Unit (degree)	Roll	Pitch	Yaw
Correct	0.6786	-89.4009	88.7162

Table 4.1: Rotational parameters of offline calibration provided by KITTI

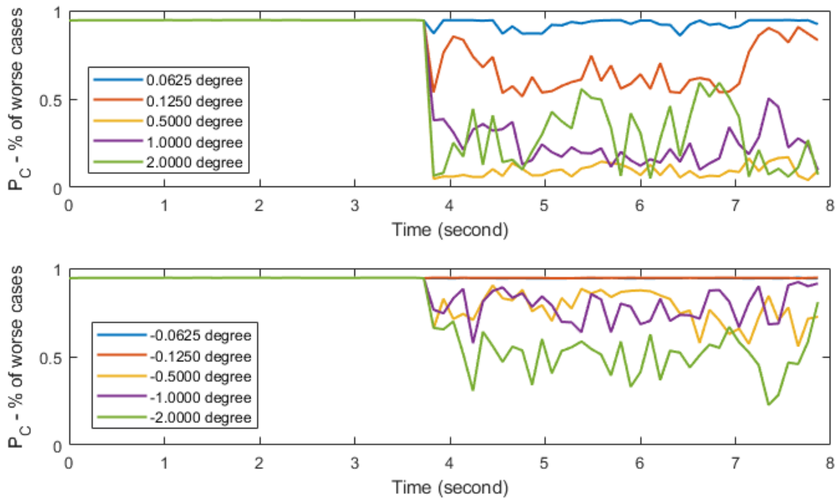


Figure 4.3: Miss-calibration detection: how P_C changes when the correct parameters are suddenly shifted by different angles A_{degree} at the time $t = 3.8285\ s$. Upper: positive A_{degree}, lower: negative A_{degree}. *(with 10 fps)*

4.3 Error variation

This section evaluates how the algorithm can continuously refine the calibration values. New experiment is done by modifying the initial correct value (see the values of group of *correct* in Table 3.1) with a shift of translation or rotation. The interval of translation shift is $[-16, 16]\ cm$, which is added on all three translation parameters (X, Y, Z). For the experiment of modifying parameters of rotation, the interval of rotation shift is $[-0.5, 0.5]°$. It is also added on all three rotation parameters $(roll, pitch, yaw)$.

4.3.1 Case of translation shift of negative $8\ cm$

The case of translation shift of negative $8\ cm$ is used to describe as an example. For the initial values of the online calibration algorithm, it is added with negative 8 cm for all three translation parameters. Figure 4.4 shows how the parameters is updated in 8 second. It starts with negative $8cm$ translation shift, and zero rotation shift at $t = 0s$. It gets a stable value around $t = 3.6s$. The result shows an effect of refinement for the initial shifted translation value, especially for X and Y directions. The absolute errors in the directions of X and Y after refinement are less than $0.04m$ $(4cm)$. For the rotation, parameter starts with correct calibration values, however, it changes slightly

		initial value	updated value	correct value	error (updated - correct)
Translation (Unit: m)	X	-0.0206	0.0216	0.0594	-0.0378
	Y	-0.1551	-0.0578	-0.0751	0.0173
	Z	-0.3521	-0.1730	-0.2721	0.0991
Rotation (Unit: degree)	roll	0.6786	0.6856	0.6786	0.007
	pitch	-89.4009	-89.4834	-89.4009	-0.0825
	yaw	88.7162	88.7305	88.7162	0.0143

Table 4.2: Calibration values with initial negative 8 cm of translation shift

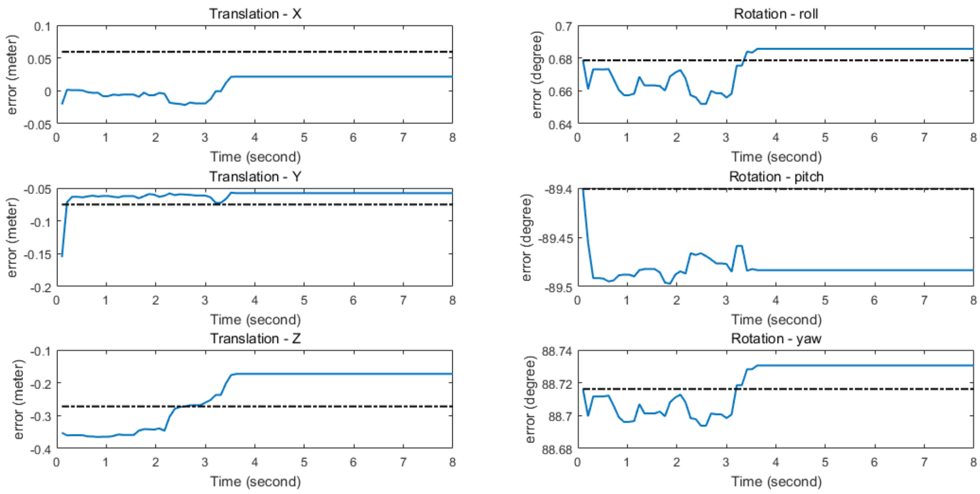


Figure 4.4: Updated calibrated values (X, Y, Z, roll, pitch, yaw) in 8 second with initial negative 8 cm of translation shift. *Dashed line indicates the correct value from the offline calibration.*

after $3.6s$ as well, which is supposed to be fixed at the initial correct values. It is because the algorithm is based on one matrix with 6 parameters, therefore, during the process of optimization by gradient descent, entire 6 parameters are being updated together and three rotational parameters have reasonably small fluctuation. The result shows the final updated rotation parameters are able to be kept around the correct in a range of less than $0.09°$. The result is satisfying. Table 4.2 shows the final values and their errors with respect to the initial and correct values. The small fluctuation of error can be clearly observed in Figure 4.5 with error bars of these six parameters. For an overall view of the error scale of translation and rotation, the root mean square error (RMSE) is used. The result is shown in Figure 4.6. The translation error after it is steady is about $0.06m$ $(6cm)$; the rotation error is about than $0.05°$.

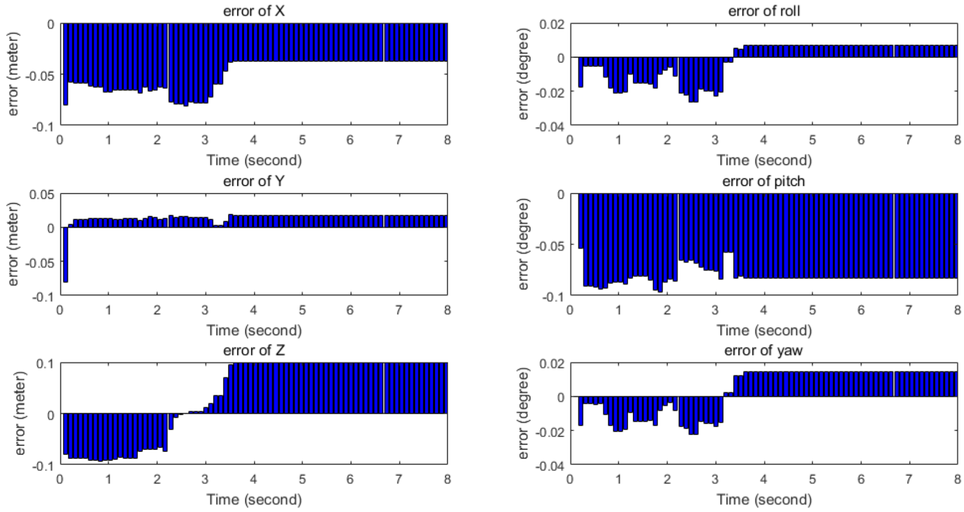


Figure 4.5: Error bar of updated calibrated values (X, Y, Z, roll, pitch, yaw) in 8 second with initial negative 8 cm of translation shift

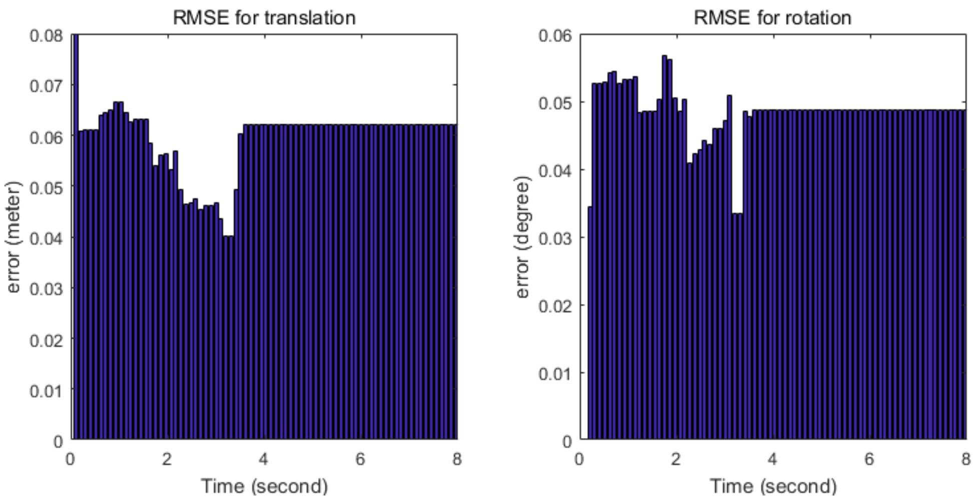


Figure 4.6: Root mean square error (RMSE) bar of updated calibrated values (translation, rotation) in 8 second with initial negative 8 cm of translation shift

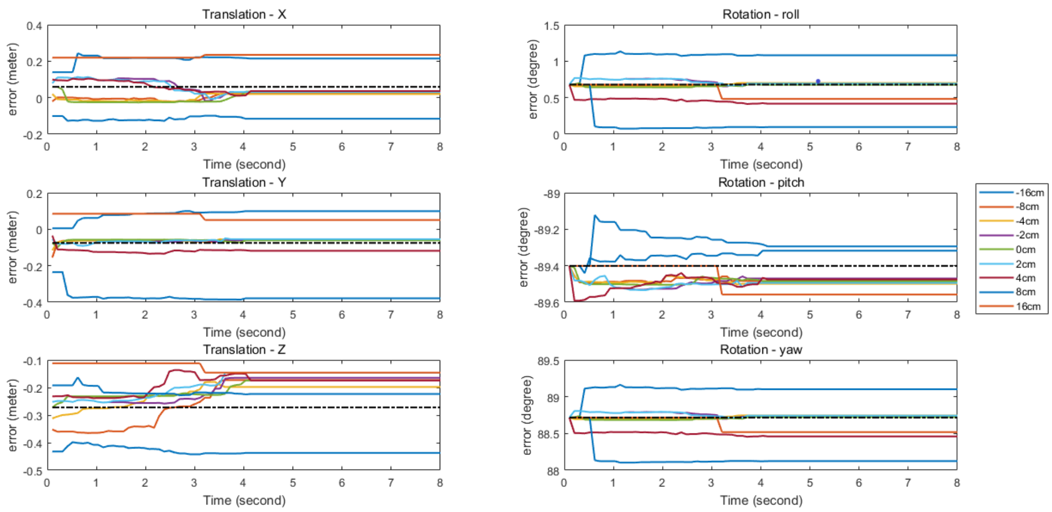
Paras	Translation shift (cm)								
	-16	-8	-4	-2	0	2	4	8	16
x(m)	0.175	0.038	0.040	0.028	0.125	0.028	0.023	0.155	0.173
y(m)	0.304	0.017	0.017	0.014	0.019	0.020	0.043	0.174	0.149
z(m)	0.165	0.099	0.074	0.107	0.047	0.126	0.097	0.049	0.150
roll(°)	0.403	0.007	0.023	0.012	0.047	0.014	0.261	0.582	0.196
pitch(°)	0.083	0.083	0.098	0.069	0.012	0.093	0.074	0.107	0.124
yaw(°)	0.385	0.014	0.030	0.020	0.037	0.026	0.256	0.592	0.198

Table 4.3: Absolute error with different initial translation shift. *Note that yellow values are regarded as outlines (whose absolute errors are larger than the mean plus one standard deviation) compared to others. It is observed that in the range of* $[-8cm, 2cm]$*, algorithm can converge calibration parameters back with a small error.*

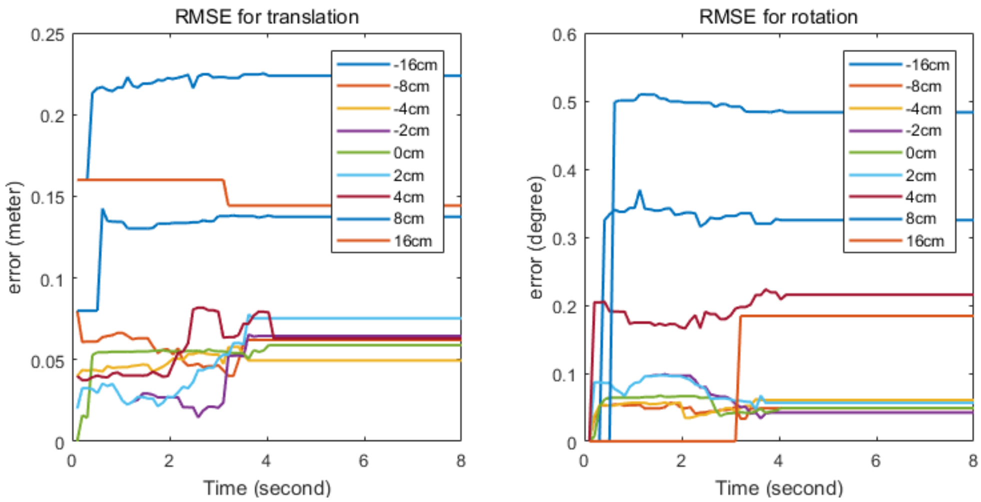
4.3.2 Comparison

To obtain the threshold of the algorithm, experiments of many different shifts are implemented. For the translation, Figure 4.7a shows the updated calibration values (X, Y, Z, roll, pitch, yaw) with different initial translation shift from $-16cm$ to $16cm$. Figure 4.7b shows the RMSE plot for translation and rotation of these different cases. After $t = 4.1s$, all parameters are in a steady state. Among these cases, the errors in the cases of $-16cm$, $8cm$, $16cm$ are obviously much larger than others, which beyond a normal range. The translation shift within $[-8cm, 4cm]$, the RMSE of translation is less than $0.0754m$, which is considered a quite small error. The translation shift within $[-8cm, 2cm]$, the RMSE of rotation is less than $0.0609°$. Hence, the range of the refinement which the algorithm is able to achieve is at least $10cm$. Table 4.3 shows the error of translation shift when the calibration is in a steady state. The outliers exist due to the limited range the algorithm can successfully converge. Since the cost function can be only considered as locally convex instead of globally, there will be a lot of optimums of calibration parameters. If the initial shift is too large, it might find other near optimal calibration parameters which are actually objectively wrong.

As for the rotation, Figure 4.8a shows the updated calibration values (X, Y, Z, roll, pitch, yaw) with different initial rotational shift from $-0.5°$ to $0.5°$. Figure 4.8b shows the RMSE plot for translation and rotation of these different cases. The updated parameters get steady at $t = 5.176$. All cases show good performance at pitch direction. The error at the pitch is less than $0.08°$ for all cases. However, for the roll and yaw, cases of $-0.5°$ and $-0.125°$, the errors



(a) Calibration parameters



(b) Root mean square error (RMSE) of calibration parameters

Figure 4.7: Updated calibrated values (X, Y, Z, roll, pitch, yaw) with different initial translation shift

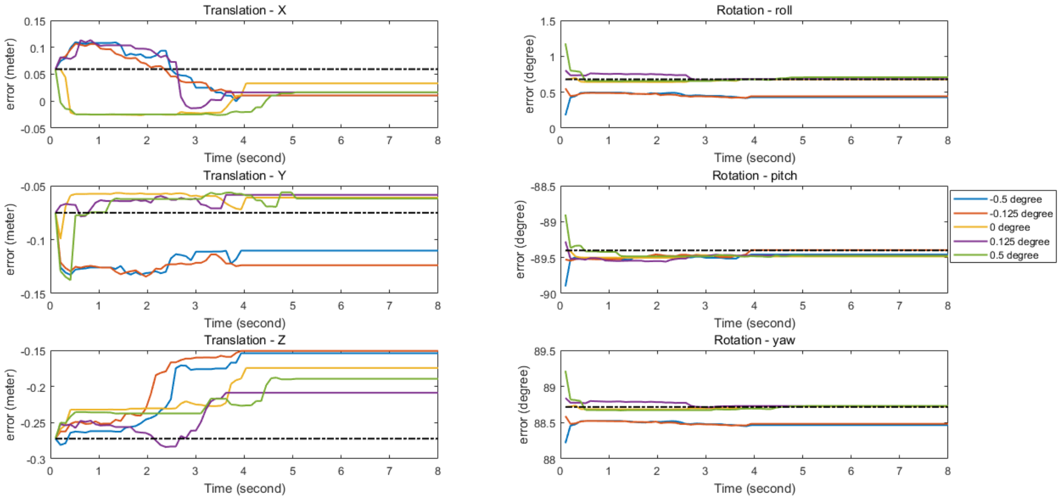
Paras	Rotation shift (degree)				
	-0.500	-0.125	0.000	0.125	0.500
x(m)	0.049	0.049	0.026	0.043	0.043
y(m)	0.035	0.049	0.014	0.017	0.013
z(m)	0.118	0.121	0.098	0.063	0.083
roll($^\circ$)	0.248	0.236	0.002	0.004	0.031
pitch($^\circ$)	0.058	0.005	0.084	0.070	0.075
yaw($^\circ$)	0.251	0.233	0.006	0.014	0.015

Table 4.4: Absolute error with different initial rotation shift. *Note that yellow values are regarded as outlines (whose errors are larger than the mean plus one standard deviation) compared to others. It is observed that in the range of* $[0, 0.5°]$*, algorithm can converge calibration parameters back with a small error.*

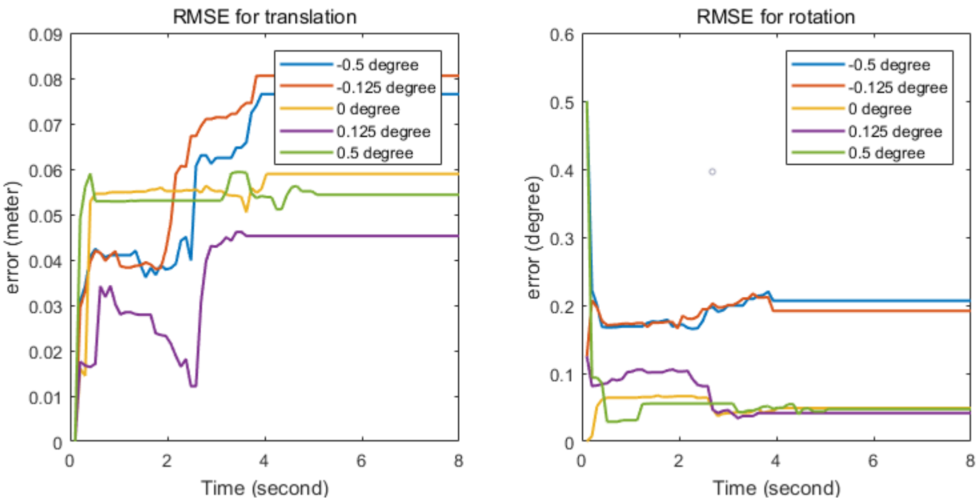
are $0.2362°$, which is 10 times larger than the value in other cases, $0.0267°$. Considering RMSE, the overall translation error is less than $0.09m$. For the rotation, it is $0.0476°$ for positive rotational shift which is about $0.5°$; for cases of $-0.5°$ and $-0.125°$, it is about $0.2066°$. Moreover, the shift whose absolute value is larger than $1°$ is also tested. It is found out that shift larger than $1°$ is out of the local optimal optimization range; the algorithm cannot update at all since both the P_C, and cost cannot keep in a high value. Table 4.4 shows the error of rotation shift when the calibration is in a steady state.

To compare with other methods, Table 4.5 shows the error of our algorithm compared to several other references [22], [21] and [28]. The methods they use are explicitly described in Chapter 2. Errors are listed from both the minimum value and average value. The translation errors are from data in the translation shift from interval $[-8cm, 2cm]$, which is able to be well online calibrated. The rotation errors are from data in the rotation shift from feasible interval $[0°, 0.5°]$. It indicates the algorithm of our system is competitive. Especially for the rotation, it shows a high accuracy; the error is near one-tenth of the error of other existing methods.

However, it should be noted that the test cases presented in the different papers may vary a lot; for instance, in terms of the driving scenarios, the interval of the refining threshold when the error is calculated, etc. Thus, this table can be used only to get an insight into the error level. [22] might test more on scenarios in old town with narrow roads and buildings on two sides, indicated from a figure of randomly selected frames in their paper. They used *the WildCat platform* including a SICK 2D Lidar and a PointGrey Bumble-



(a) Updated calibrated values (X, Y, Z, roll, pitch, yaw) with different initial rotation shift



(b) Root mean square error (RMSE) of calibration parameters

Figure 4.8: Updated calibrated values (X, Y, Z, roll, pitch, yaw) with different initial rotation shift

Paras	[22]	[21]	[28]	Our system	
				min	average
X(m)	0.0045	0.02	0.305	0.028	0.052
Y(m)	0.0052	0.014	0.005	0.014	0.018
Z(m)	0.0046	0.006	0.446	0.047	0.091
roll($^\circ$)	0.38	0.672	0.15	0.026	0.037
pitch ($^\circ$)	0.39	0.628	0.00	0.013	0.015
yaw ($^\circ$)	0.44	0.476	0.27	0.063	0.081

Table 4.5: Absolute error of online calibration result compared with different methods. *Note that all values are the absolute values.*

bee stereo camera. [21] shows more similarities for the testing scenarios from their figures. The dataset they used was collected on a normal city road in a late afternoon. Some images suffered from *lens flare, ghosting, and blooming, underexposure*. For the sensors, they also used a Velodyne HDL-64E Lidar and a PointGray Ladybug3 Panoramic camera. [28] had the same equipment of [21]. Their result is based on the window size of 40 scans. They tested both indoor environment and outdoor like a parking lot.

4.4 Impact of acceleration on P_C

In the Section 3.3, P_C, the percentage of worse cases, is used to check the correctness of the calibration parameters. More accurate parameters will give a higher P_C. However, P_C is severely influenced by the acceleration. Figure 4.9 shows how P_C changes in a 96-second long video with correct calibration parameters. Here velocity and acceleration data are provided by IMU, whose direction is forward respect to the vehicle body. To have the correlation among these three values using the same scale, both velocity and acceleration are normalized by their maximum value, so that the maximum scale to be 1. From the figure, if the absolute value of acceleration is larger than $0.135 (= 0.07 * 1.9215)\ m/s^2$, it is observed that P_C is hard to keep in a high percentage value even it is with correct calibration parameters. It might be due to Lidar data distortion by the motion of the vehicle. Therefore, in the algorithm, P_C is not the only one criteria for updating calibration parameters, but the cost value is also taken into consideration.

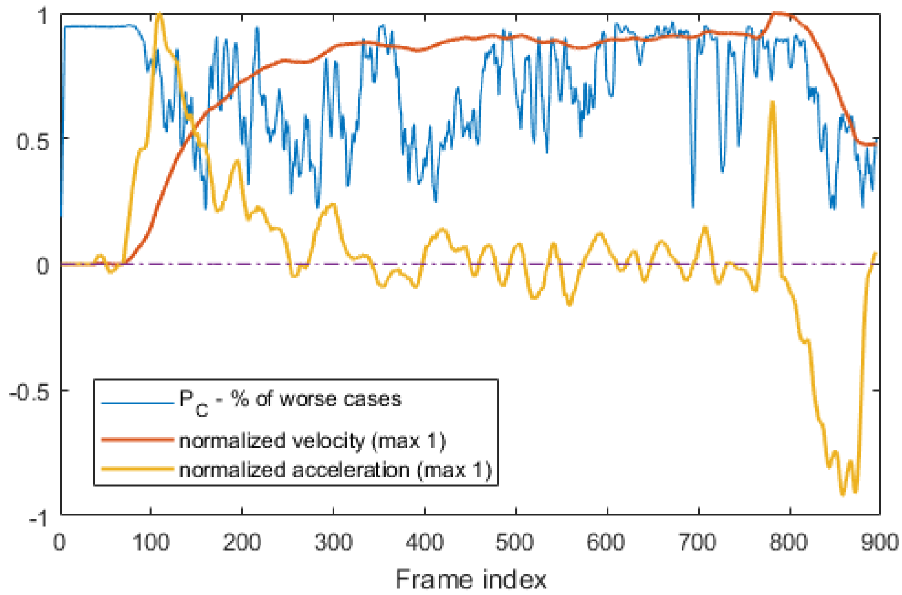


Figure 4.9: Correlation of acceleration and worse percentage P_C. *Note: the data of both velocity and acceleration are normalized by their maximum (which are* $18.5119\ m/s$ *and* $1.9215\ m/s^2$ *respectively).*

Chapter 5

Conclusions

The master thesis project is about the online calibration between the Lidar and camera sensors. It presents a solution to find the calibration parameters, (X, Y, Z) for translation, $(roll, pitch, yaw)$ for rotation between the coordinates of two sensors. The project researches on miss-calibration detection. The experiment shows it can successfully detect the miss-calibration in less than 0.0005s, when the translation and rotation shifts are larger than $12cm$ and $0.6250°$ respectively. The algorithm also can continuously refine the parameters during the vehicle motion in a certain threshold interval. It is capable to correct the parameter with at least $10cm$ shift for translation and $0.5°$ for rotation from the correct calibration parameters. Compared with other existing methods, it is competitive. It even has better results on rotational parameters. The method we added to cluster the point cloud during the feature extraction, can increase the accuracy when to find the correspondence. Moreover, the algorithm can be run in real-time.

5.1 Future work

To solve the Lidar motion distortion, many solutions can be applied, like multiple variants of Iterative Closest Point method (ICP) that estimate the Lidar velocity can be used for correction.

For the miss-calibration algorithm, it would be interesting to research on the performance of it for each specific parameter. For example, only translation shift on X, or only rotation shift on roll etc.

Bibliography

[1] *J3016TM Levels of Driving Automation*. SAE International, Dec, 2018.

[2] I. Harner, "The 5 autonomous driving levels explained."

[3] J. Cohen, "Ai... and the vehicle went autonomous."

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," 2012.

[5] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," *IEEE International Conference on Robotics and Automation(ICRA)*, 2015.

[6] "Velodyne hdl-64e data sheet."

[7] B. L. H. W. Markus Maurer, J. Christian Gerdes, *Autonomous driving*. 2016.

[8] A. Maxmen, "Self-driving car dilemmas reveal that moral choices are not universal."

[9] R. P. Qilong Zhang, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[10] R. S. Huijing Zhao, Yuzhong Chen, "An efficient extrinsic calibration of a multiple laser scanners and cameras," *IEEE Intelligent Vehicles Symposium*, 2007.

[11] M. B. HLili Huang, "A novel multi-planar lidar and computer vision calibration procedure using 2d patterns for automated navigation," *IEEE Intelligent Vehicles Symposium*, 2009.

[12] C. B. S. Andreas Geiger, Frank Moosmann, "Automatic camera and range sensor calibration using a single shot," *ICRA*, 2012.

[13] J. I. N. Zachary Taylor, "Motion-based calibration of multimodal sensor arrays," *ICRA*, 2015.

[14] D. M. Carlos Guindel, Jorge Beltran, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," *IEEE*, 2017.

[15] L. D. X. C. D. Z. Ganhua Li, Yunhui Liu, "An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3854–3859, October 2007.

[16] C. S. W. K. C. K. S. S. Yoonsu Park, Seokmin Yun, "Calibration between color camera and 3d lidar instruments with a polygonal planar board," *Sensors, 14(3):5333–5353,2014*, p. 14(3):5333–5353, 2014.

[17] D. . B. H. . K. T. Kwak, K. ; Huber, "Extrinsic calibration of a single line scanning lidar and a camera," *IEEE*, pp. 3854–3859, 2011.

[18] B. B. Hatem Alismail, L. Douglas Baker, "Automatic calibration of a range sensor and camera system," *2nd International Conference on 3D Imaging, Processing, Visualization and Transmission (3DIMPVT)*, 2012.

[19] L. H. Zoltan Pusztai, "Accurate calibration of lidar-camera systems using ordinary boxes," *ICCV*, 2017.

[20] R. Z. Peyman Moghadam, Michael Bosse, "Line-based extrinsic calibration of range and image sensors," *IEEE*, 2013.

[21] S. T. Jesse Levinson, "Automatic online calibration of cameras and lasers," *Robotics: Science and Systems*, 2013.

[22] P. N. Ashley Napier, Peter Corke, "Cross-calibration of push-broom 2d lidars and cameras in natural scenes," *ICRA*, 2013.

[23] N. S. Blaga, Bianca-Cerasela-Zelia, "Online cross-calibration of camera and lidar," *13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 295–301, September 2017.

[24] E. E. H.-P. S. Lionel Baboud, Martin Čadík, "Automatic photo-to-terrain alignment for the annotation of mountain pictures," *CVPR*, 2011.

[25] S. T. Jesse Levinson, "Unsupervised calibration for multi-beam lasers.," *International Symposium on Experimental Robotics*, 2011.

[26] R. S. Ashley Napier, "Robust maximum-likelihood on-line lidar-to-camera calibration monitoring and refinement," *23rd Computer Vision Winter Workshop*, February 5-7, 2018.

[27] J. K. Seungpyo Hong, Heedong Ko, "Vicp: Velocity updating iterative closest point algorithm," *IEEE International Conference on Robotics and Automation*, 2010.

[28] S. S. Gaurav Pandey, James R. McBride, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics. Volume 32 Issue 5.*, pp. 696–722, August 2015.

[29] J. F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[30] "Eroding and dilating. opencv document. image processing (imgproc module).."

[31] J. M. B. Jonathan Barzilai, "Two-point step size gradient methods," *IMA Journal of Numerical Analysis, 8*, pp. 141–148, 1988.

[32] S. S. R. E. Gaurav Pandey, James McBride, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics. 32. 10.1002/rob.21542*, 2014.

[33] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

TRITA -EECS-EX-2019:760