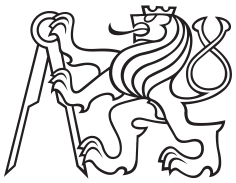
Bachelor Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Lidar Pose Calibration Using Coded Reflectance Targets

Matej Novosad

Supervisor: Ing. Martin Matoušek, Ph.D.
Field of study: Robotic Perception
August 2021

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 11th August 2021

Abstract

As autonomous cars gain popularity, so does interest in what are thought to be the best sensors. Currently those are LiDARs and cameras. Having multiple sensors on the same construction requires precise calibration. We present reflective coded targets used for calibration. This thesis presents the whole process of development from target design to testing.

Keywords: LiDAR, calibration, target, detection

Supervisor: Ing. Martin Matoušek, Ph.D.

Abstrakt

Jak samořídící auta získávají na popularitě, tak roste in zájem o senzory, jež jsou považovány za nejlepší pro danou problematiku. Momentálně jimi jsou LiDARy a kamery. Mít více těchto senzorů na stejné konstrukci vyžaduje přesnou kalibraci. My představujeme odrazné značky jako kalibrační pomůcku. Tato práce obsahuje celý vývoj od návrhu značky po její testování.

Klíčová slova: LiDAR, kalibrace, značky, detekce

Překlad názvu: Kalibrace polohy lidaru pomocí kódovaných odrazných značek

Contents

1 Introduction **1**

2 Calibration using reflective targets **3**

2.1 LiDAR data 3

2.2 Target design 4

2.3 Code implementation 5

2.3.1 Data structure 5

2.3.2 Algorithms 6

3 Experiments and testing **13**

3.1 Testing 13

3.1.1 First target design 13

3.1.2 Second target design 15

4 Conclusion **21**

A Bibliography **23**

B Project Specification **25**

Figures

2.1 First target design 4

2.2 Example of the second target design 5

3.1 Measurement setup for the first target 14

3.2 Frequency of target detection based on distance 15

3.3 Target reflectivity graph comparison when target is further (left) and closer (right) 15

3.4 Target detection results 16

3.5 Target detection results 17

3.6 Calibrated coordinate systems . . 18

3.7 Calibrated coordinate systems . . 19

Tables

3.1 Detection table for target 1 16

3.2 Detection table for target 2 17

Chapter 1

Introduction

As autonomous cars gain popularity and keep getting more and more attention from both public and engineers, certain sensors prove to be more applicable than others. Currently, most popular sensors in this field are cameras and Light Detection and Ranging (LiDAR). It is a common practice to use multiple of those sensors on the same vehicle, but this requires a proper calibration of all those sensors. Many di□erent methods and objects are used for calibration process. Objects used are both planar and spatial.

For example, Pusztai et al. [1] used a simple cardboard box utilizing its accessibility, Gao and Spletzer [10], on the other hand, used pieces of retro-reflective tape upon vertical poles. Geiger et al. [5] decided to go with planar chessboards. They only needed one shot for their calibration, however, it did require multiple chessboards. Interesting solution was presented by Velas et al. [4], who proposed planar markers as di□erent shapes cutouts in front of white background. Their small downside, however, is the assumption that the dimensions of objects are known beforehand. Park et al. [3] also went for a object without any texture, just a clear white object in a shape of a triangle or rhombus. Main strength of this method is that there is no additional LiDAR noise created by textures.

This thesis introduces coded reflective targets. This allows calibration to be setup practically anywhere, since all you need is a planar target printed on a piece of paper. Also, this type of a target is easily detectable by both LiDARs, thanks to its ability to detect reflectivity of the surface, and camera. Camera obviously provides really high-resolution color images, so it should have no problem detecting color-based target. That is why the main focus is

on LiDARs. This will be an o□ine calibration, meaning that the calibration is done once, before usage of the sensoric system.

Chapter 2

Calibration using reflective targets

The goal is to calibrate LiDARs using a coded target. This means that there could be more of them and all are distinguishable from one another by their code. Since density of LiDAR data is much higher in one direction, code would obviously be much easier to read in that direction too. We decided to go for three intensity levels on the target: black white and gray. This would allow for an ability to design more targets with unique codes, while also introducing some limitations to reduce false identifications.

Calibration would work as follows. Targets are to be placed around the area. Multiple scans are performed to ensure precision of calculations. From measured data, all the targets measured at the same time, with the same code, by at least two di□erent sensors would be recognized as the same and their coordinates would be use to calculate transformation matrix between the two coordinate systems.

2.1 LiDAR data

LiDAR (light detection and ranging) is an optical remote-sensing technique that uses laser light to densely sample the surface of the earth, producing highly accurate x,y,z measurements.

LiDAR scans the area to record its data points and form a point cloud.

The sensor itself is composed of two parts: laser emission and laser reception. The emission system works by leveraging layers of laser beams. The more layers, the more accurate the sensor is. The LiDARs we used had 16 layers. The precision of measurement within each layer is set beforehand by adjusting angle difference between two pulses. The smaller the difference, more data will be recorded in a single scan resulting in higher precision.

Every recorded point is defined by its Cartesian coordinates, reflectivity, layer and angle at which it was recorded. We will attempt to use reflectivity parameter to distinguish transitions between two intensity levels of the target, therefore being able to read the code written on the target.

2.2 Target design

For target's first design, a rectangle shape was selected. The rectangle was divided into 7 vertical stripes of either black, white, or grey color with the following rules: first and last stripe are always black, second and second to last are always white. This is to mark the beginning and the end of the target, limiting any potential false identifications. Also, the third and fifth stripe couldn't be white, so the first and last two stripes could be used to estimate length of each section of certain intensity level. Visualization of color distribution can be seen in Figure 2.1. This would leave us with 12 possible different variations of the target. The downsides of this design and why we decided to change it will be explained later on, in chapter 3.

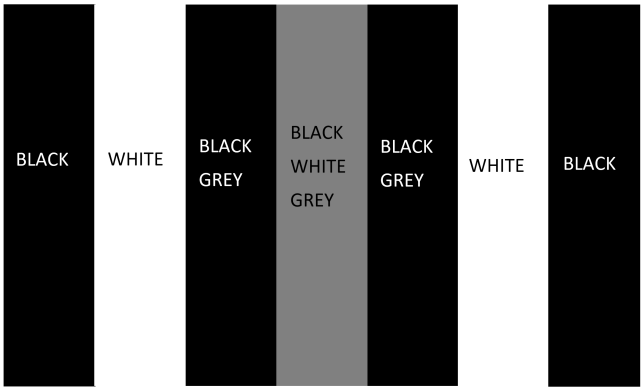


Figure 2.1: First target design

For the second design we decided to go with a triangle. This allowed us to find the exact position of a certain point (since it is a triangle this was be the top point of it) just by detecting the target with two laser layers and the most

left and right points will create two lines. The intersections of these two lines are the top point of the triangle. Additional adjustment was to never have the same intensity level in two adjacent stripes. Therefore, we are always able to precisely detect transitions between stripes making it the top point estimate more accurate. So, the target is basically four isosceles triangles that have bases of di□erent lengths, but laying on the same line and the third point is the same for all four triangles. New rules about stripes coloring made it possible for only 6 di□erent targets to meet the requirements, but that is more than enough, because, in theory, we only need one target found multiple times. However, having more of them is only a bonus that makes it possible for more points to be found, therefore ensuring better quality of the result. An example of the second design target is displayed in Figure 2.2. The expected downside to this target compared to the first design was that it would make it harder to detect the same target in more than three layers within one measurement, as the length of each stripe is smaller the closer to the top it is. However, to be able to find a line, we only need to detect it two times.

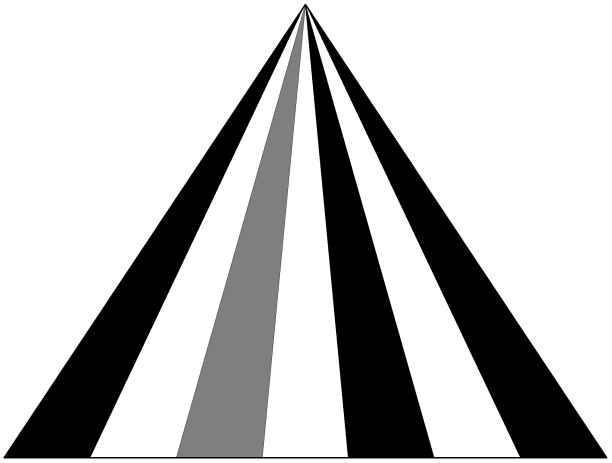


Figure 2.2: Example of the second target design

2.3 Code implementation

2.3.1 Data structure

Each measurement is series of points as one frame divided into 16 laser layers representing one row. Each point is an object with following parameters:

- Cartesian coordinates (x, y, z) which of course represent its location relative to sensor
- reflectivity – main focus of the thesis, using this parameter we try to distinguish colors, or precisely in this case white, grey and black
- laser ID – defining in which layer given point was measured
- channel ID – basically an angle of the point in cylindrical coordinates, defines order within a single laser measurement
- time – time at which a given point was measured

2.3.2 Algorithms

When we read the file containing the data for each one measurement, we sort it into a 2D array of previously described object point. This array is divided into rows depending on each point's laser ID and each row is sorted based on channel ID. Not all rows are the same length due to some points being marked as invalid.

Start detection

```
firstBlack = None
firstWhite = None
detected = list() for i=1 to len(row)
    if firstBlack = None AND row[i] = black
        firstBlack = row[i]
    if firstBlack != None
        if firstWhite = None AND row[i] = white
            firstWhite = row[i]
            blackLength = distance(firstBlack, firstWhite)
        if firstWhite != None
            if row[i] != white
                border1 = row[i]
                whiteLength = distance(firstWhite, border1)
                if abs(blackLength-whiteLength) < allowed
                    expectedLength = average(blackLength, whiteLength)
                    detected.append([firstBlack, firstWhite, border1])
                else
                    firstWhite = None
```

```
firstBlack = None
```

Each row is checked individually at first looking for a valid code. The first stripe of the target is always black and the second one is always white. Therefore, we check each for all sequences of black followed by a sequence of white of the same length with the following rules. If a point's reflectivity is below a given threshold for black, we mark that point as a candidate for the first black point of the target. Then each following point is checked until its reflectivity suggests it is no longer black. Distance between first point and the last one is saved as expected length. Then a percentage of this length is allowed for the next couple of points to reach threshold for white and once again we repeat everything from the previous step except this time end cause is when the points reflectivity is no longer considered as white. During this whole process if distance between two points that are supposed to be next to each other is bigger than defined value, the calculation stops, candidate point is removed, and the algorithm continues looking for a black point again. If, however, all the steps above are successful overall length of detected black stripe is compared to the length of the white stripe and if they are similar enough, detected section is marked as possible beginning of the target and saved.

Read Code

for every detected start point:

```
index = border1
 for code=1 to 3
    intensity = list()
    for i = index to index+expectedLength+tolerance
       intensity.append(row[i].reflectivity)
       if intensityDi□erence(row[i], row[i-1]) > value
          border = row[i]
          length = distance(border, lastBorder)
          if abs(length-expectedLength) < allowed
             break
          else
             return FAIL
    reflectivity = average(intensity)
    if reflectivity = black
       save as black
    else if reflectivity = white
```

```
        save as white
    else
        save as grey
```

Once we get all potential first two stripes, we take each pair try to read a code in the following three stripes. We save reflectivity of each point until the difference in reflectivity between two adjacent ones is greater then defined value. Then we check the length of the stripe and compare it to expected length calculated from the length of first and second stripe. If it is within acceptable margins, we take the saved reflectivity values and calculate average for the given section. Depending on average value we assign it one of the possible colors, based on both predefined values and black and white values calculated from the first two stripes. This has to be done because, logically, the further away the target is, lower the reflectivity value is for both white and grey. We do this three times, for three middle stripes. Every single point of transition between two stripes is saved.

End detection

```
i = lastBorder
while row[i] = white AND distance(lastBorder, row[i]) <= allowed
    i=i+1
lastWhite = row[i]
if distance(lastBorder, lastWhite) > allowed
    return FAIL
while row[i] = black AND distance(lastWhite, row[i]) <= allowed
    i=i+1
lastBlack = row[i]
if distance(lastBlack, lastWhite) > allowed
    return FAIL
else
    target is found
```

If all of the lengths are acceptable, we move to the final detection step, where we need to detect white and black stripe to confirm that this was in fact the target. This is done similarly to the first part, where we look for black, then white. Now its just in reverse. However, if length of the stripe is not in expected margins we discard an entire target, at least for the given row. If on the other hand, detection is successful, it is saved into a dictionary with the key being its code and value an array of points representing borders

between each individual stripe.

Removing possible misidentifications

As this is done for every single row, dictionaries are merged, making the value of each key be an 2D array of detected border points.

This dictionary first has to be checked for any misidentifications. Since there is always only of each target in a single measurement, it is expected that the detected sections should be just above each other, becoming narrower the higher up they are. If some are, but others aren't, the biggest clump is considered to be correct.

If there are sections with the same code just above one another, they are checked again to determine if there could have been a misidentification in just one stripe in a single section, usually black mistaken for grey in case of a bigger distance. If this isn't the case, this target is removed. Target is also removed if there are no sections detected closely above it, since we need to detect target in at least two lasers in order to be able to find the top of the triangle.

Finding top point of triangle

Once all of the rows were read and any false identifications were removed, we move to detecting the top point of the triangle. So first all of the points we detected as a part of the target are attempted to fit into a plane with the smallest error. This is done using the best fit function that uses the singular value decomposition. If it really is the target all of the points should fit with really small errors caused by only noise, since the target itself is planar. If every point is close enough to a found plane, we proceed to project each of the point onto the plane. The first guess for the top point is then calculated from the points of left and right margin of the target in following way: we fit a line through all of the left points and do the same for the right. If there are only two points, which was often the case, then there is only one line that passes through both. If there were more then two, best fit function was used again to find the line. Intersections of these two lines became the first estimated top point.

This point is then slightly adjusted to best fit remaining lines that are should go through margins between stripes and then intersect in the top point. Every top point is assigned time when it was measured taking average time of each point that is part of the target. In the end, every detected target is assigned coordinates of its top point, code and time.

Transformation matrix

Every found top point is than compared to all the others. If two targets have the same code and were measured within 5 milliseconds of each other, they are considered to be the same point at the same point in time, therefore making it a valid pair for calculation of transformation matrix. Transformation matrix is calculated using all of the found pairs of valid points.

Transformation matrix is a matrix that transforms homogeneous coordinates of every point in one coordinate system into homogeneous coordinates of the second one and has a following structure:

$$T = \begin{pmatrix} a_{11} & a_{12} & a_{13} & d_1 \\ a_{21} & a_{22} & a_{23} & d_2 \\ a_{31} & a_{32} & a_{33} & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This matrix should satisfy the following equation:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & d_1 \\ a_{21} & a_{22} & a_{23} & d_2 \\ a_{31} & a_{32} & a_{33} & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

where x, y, z are point's coordinates in coordinate system of the first LiDAR and x', y', z' are coordinates in coordinate system of the second LiDAR.

In order to calculate this matrix, we need to find the parameters a_{ij} and d_k for i, j, k {1, 2, 3}. This was done using all of the found corresponding point pairs. We sort all of the points into two matrices A and B.

$$A = \begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \quad B = \begin{pmatrix} x'_1 & y'_1 & z'_1 & 1 \\ x'_2 & y'_2 & z'_2 & 1 \\ x'_3 & y'_3 & z'_3 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Where (x_n, y_n, z_n) and (x'_n, y'_n, z'_n) are a pair of points corresponding to one another as explained above. Now all left to do is solve the matrix equation

$$A \cdot T^T = B \tag{2.1}$$

$$T^T = pinv(A) \cdot B \tag{2.2}$$

$$T = (pinv(A) \cdot B)^T \tag{2.3}$$

and so we calculated the transformation matrix between coordinate systems of two LiDARs.

Chapter 3

Experiments and testing

3.1 Testing

3.1.1 First target design

Two valid targets were printed out and placed into measurement area. The setup can be seen in Figure 3.1. Smaller data set of 41 measurements was created containing data with these two targets. Immediate takeaway was that, as is common for most measurements, the further the LiDAR is to the target, logically, less times does the code detect the same target. This proves to be troublesome for targets that are only detected with one laser, as it would be impossible to correctly identify its position.

In Figure 3.2 are the results for how many lasers detected the same target and that target's average distance from the sensor calculated as an average distance of each detected point belonging to the target. Clearly, we can see that for targets that are furthest away, only one laser detected the target, but in most cases there are multiple. For the closest targets, success rate is way better, peaking at detecting the target with six lasers. Distance to target a□ects not only the number of lasers in which the code is able to identify the target, but also quality of reading as seen in comparison of reflectivity data for the same target from di□erent distances in Figure 3.3. These graphs show the readings of lasers that both detected the same target, however at di□erent distance. Most noticeable di□erence is the scale of deviations around

Figure 3.1: Measurement setup for the first target

of reflectivity between points next to each other that are supposed to be same. While for closer distance those deviations are at most around 5 %, for target further away its close to 20 % for worst cases, most frequently for white stripes. We can also notice pointy peaks for white stripes which make it di□cult to recognize if we are reaching transition between two colors. Transitions are also much cleaner for closer targets being much closer to step function then its further away counterpart which is takes a shape more similar to a letter “U”.

Quality of detection algorithm is presented in Figure 3.4. It also furthermore proves the point of detection quality based on distance as seen by comparing number of rows that the target was detected in and length of target in channels, which is logical since it has much more data to process so it is„ of course more precise. Color of each point is determined by its reflectivity, higher reflectivity means lighter color. Red marks are detected transitions between each stripe as well as beginning and end of the detected target. Of course, only detail of data where the target was detected is displayed.

Even though this target proved to be good to test out algorithm’s ability to read the coded part of the target, which works pretty well as proved by figures above, it is not the final solution to the given problem, because we are only able to identify transitions between stripes but aren’t able to successfully

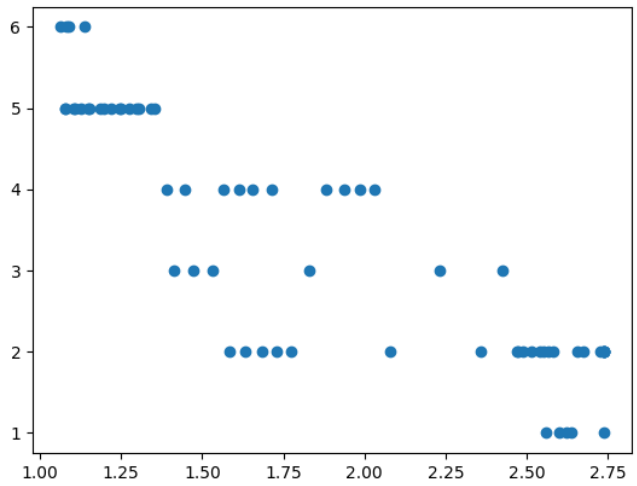


Figure 3.2: Frequency of target detection based on distance

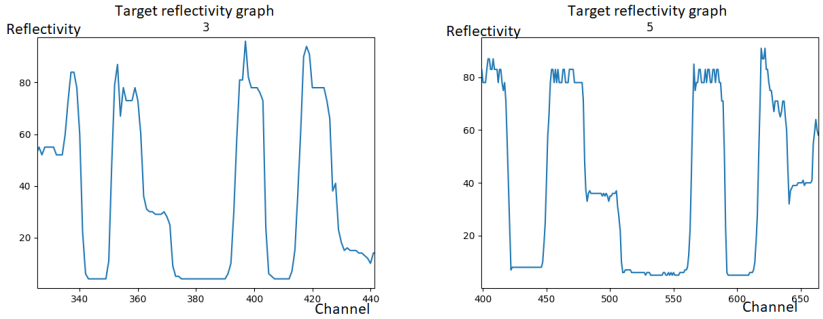


Figure 3.3: Target reflectivity graph comparison when target is further (left) and closer (right)

identify corners or any other points from which we could know the exact position of the target. That is why the second design was created.

3.1.2 Second target design

As this was to be the final product much more data was recorded. 1444 measurements for two different LiDARs, with the ultimate goal of finding enough pairs of top points of target triangle that were recorded at almost the same time, within a given time frame. For 1444 measurements detection success is displayed in TABLES 3.1 and 3.2. Unfortunately we can't really talk about success rate, since as deducted from results when attempting to detect the first generation target, for too great of a distance, target cannot

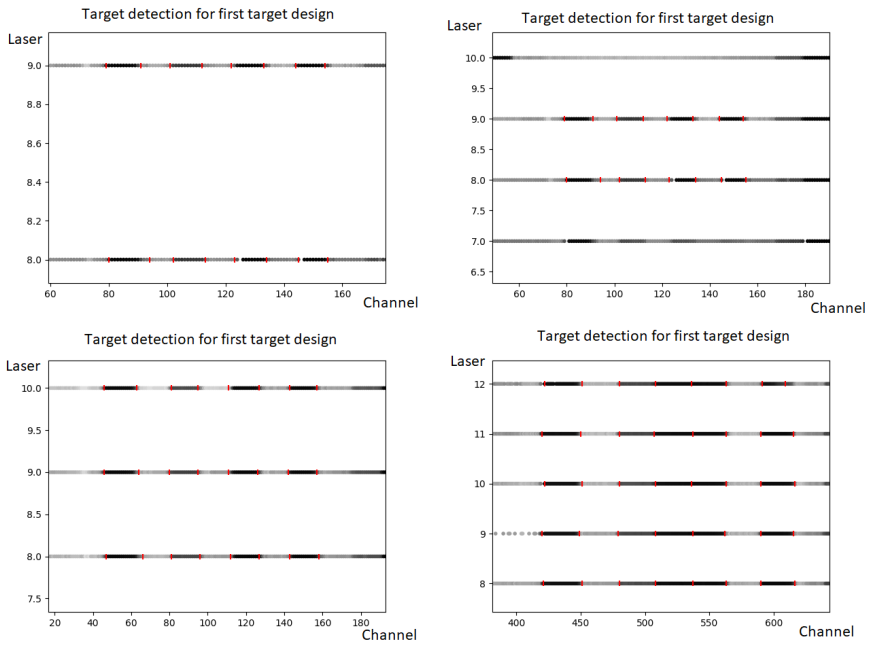


Figure 3.4: Target detection results

LiDAR ID	detected by one laser	detected by multiple lasers
LiDAR 1	373	591
LiDAR 2	237	430

Table 3.1: Detection table for target 1

be detected successfully. Furthermore, the data set is too great to check each file manually. This is also the reason why we decided to go with dynamic programming in the first place, rather than AI.

Because of the huge amount of data, angle between two points was doubled, resulting in half the points per row compared to the previous measurements for rectangle target. As seen in figure 3.5 some of the points belonging to the target itself are missing due to sensor reading them as invalid, quite frequently right at the margin of stripes. All this increased the amount of both undetected targets and false identifications at first, so small adjustment were to be made.

Regardless of those complications, we were able to detect targets in many cases, as seen in tables 3.1 and 3.2. In three of four cases the target was even detected by more then one laser in third of all recorded data, which is important since it is crucial for finding the top point of the triangle.

LiDAR ID	detected by one laser	detected by multiple lasers
LiDAR 1	317	261
LiDAR 2	315	475

Table 3.2: Detection table for target 2

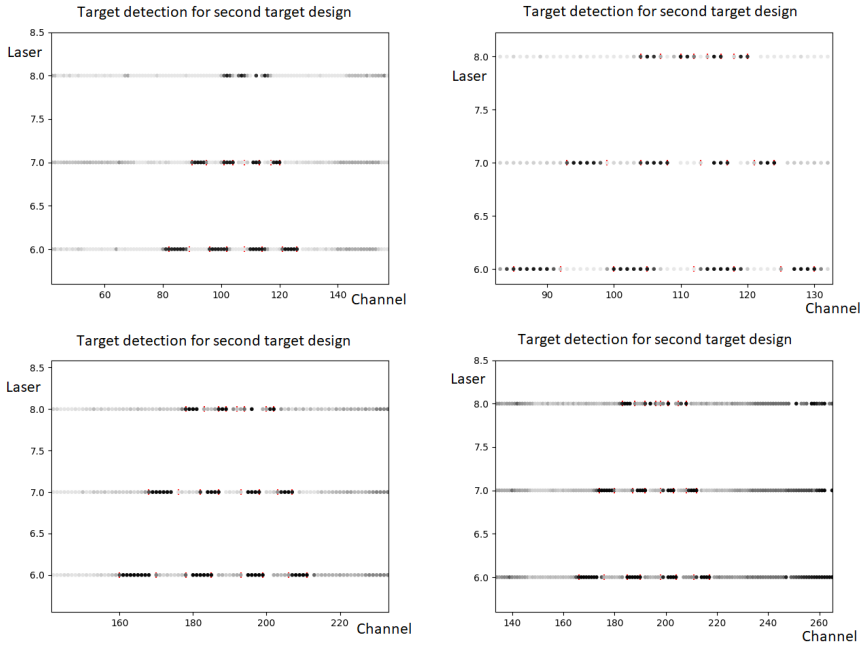


Figure 3.5: Target detection results

After all the top points were calculated and compared to each other, 41 pairs of points were found that meet the requirements. This gave a solid amount to calculate the transformation matrix and finish the calibration.

In figures 3.6 and 3.7 we can see results of the calibration between two LiDARs. Red dots are points in the coordinate system of the first LiDAR and blue ones are transformed coordinates of points measured by the second LiDAR. We can clearly see that the walls align with minimal error. Therefore calibration can be considered successful.

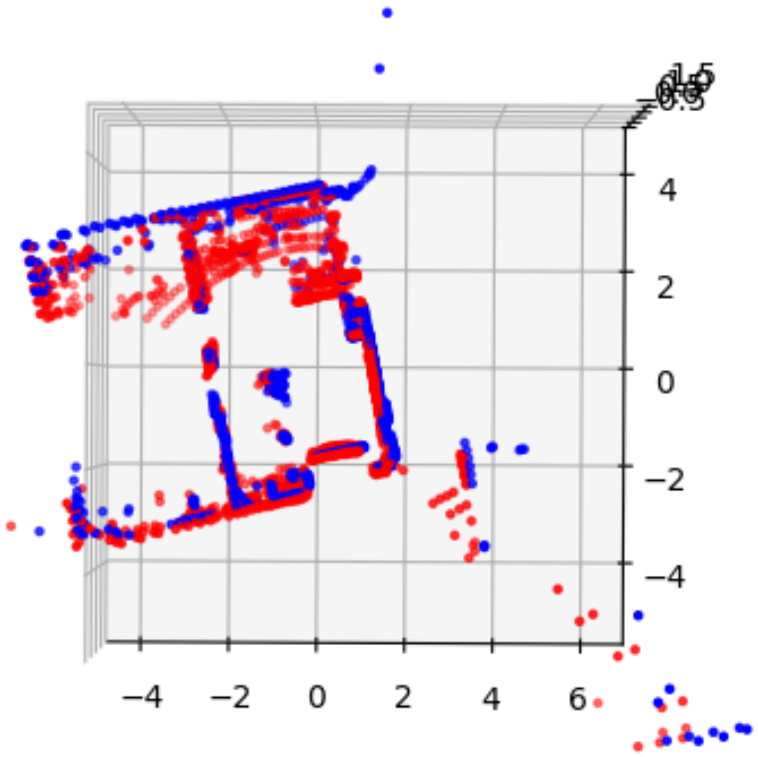


Figure 3.6: Calibrated coordinate systems

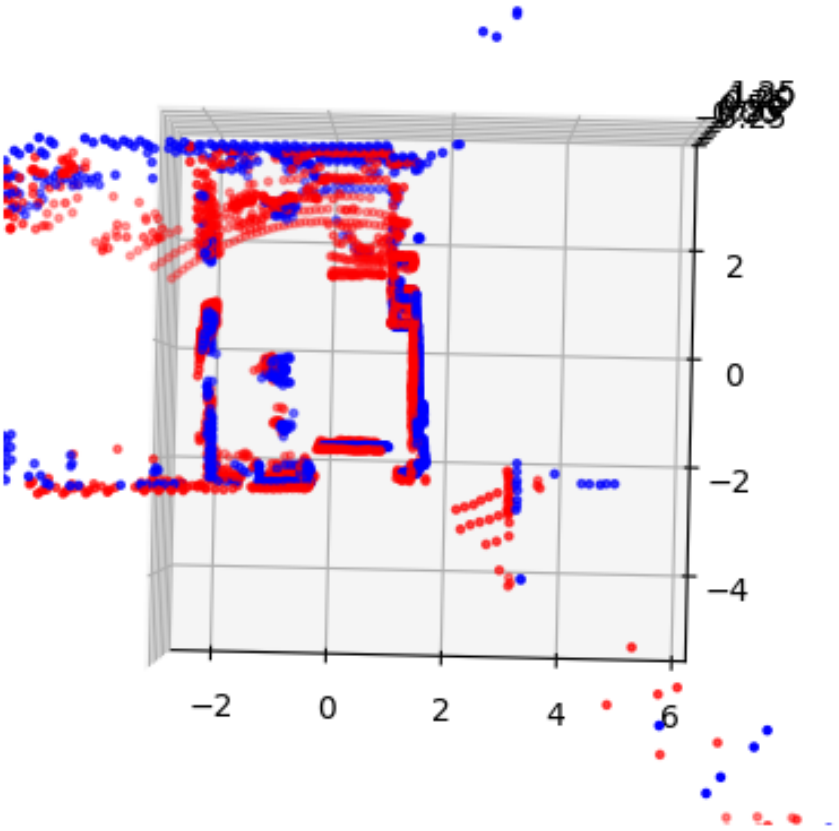


Figure 3.7: Calibrated coordinate systems

Chapter 4

Conclusion

In this thesis, we proposed special markers for LiDAR to LiDAR calibration, however in theory it might be applicable for cameras, too. The calibration using this method was ultimately successful, being able to align area around sensors accordingly. However, there is certainly room for improvement, mostly in optimization methods for top of the triangle detection.

Also, distance from target to sensor created a lot of problems, so a di□erent approach to measurement process might be better. For example to always make sure that the targets are close to the sensors, rather than just scanning the area randomly. That would result in more targets being detected and also more precise. Therefore more points could co-respond considering the time they were measured so the transformation matrix would be more accurate, too.

Appendix A

Bibliography

[1] Pusztai, Eichhardt, Hajder, Levente. (2018). Accurate Calibration of Multi-LiDAR-Multi-Camera Systems. Sensors. 18. 2139. 10.3390/s18072139.

[2] Alismail, H.; Baker, L.D.; Browning, B. Automatic calibration of a range sensor and camera system. In Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission, Zurich, Switzerland, 13–15 October 2012; pp. 286–292.

[3] Park, Y.; Yun, S.; Won, C.S.; Cho, K.; Um, K.; Sim, S. Calibration between color camera and 3D LIDAR instruments with a polygonal planar board. Sensors 2014,14, 5333–5353.

[4] Vélas, M.; Španěl, M.; Materna, Z.; Herout, A. Calibration of RGB Camera With Velodyne LiDAR. In WSCG 2014 Communication Papers Proceedings; Union Agency: Plzen, Czech Republic, 2014; Volume 2014, pp. 135–144.

[5] A. Geiger, F. Moosmann, Ö. Car and B. Schuster, "Automatic camera and range sensor calibration using a single shot," 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 3936-3943, doi: 10.1109/ICRA.2012.6224570.

[6] Hartley, Richard and Zisserman, Andrew – Multiple View Geometry in computer vision – Cambridge university press, 2003.

[7] Farouk Ghallabi, Fawzi Nashashibi, Ghayath El-Haj-Shhade, Marie-AnneMittet–LIDAR-BasedLaneMarkingDetection For Vehicle Positioning in an HD Map – 21th IEEE Intl. Conf. on Intelligent Transportation Systems, Nov 2018, Maui, Hawaii United States

[8] Hata, A. Wolf, D. –Road marking detection using LIDAR reflective intensity data and its application to vehicle localization - 17th IEEE Intl. Conf. on Intelligent Transportation Systems, Qingdao, 2014

[9] Zhou, L. and Li, Z. and Kaess, M. – Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences – In Proc. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, Oct. 2018

[10] C. Gao and J. R. Spletzer, "On-line calibration of multiple LIDARs on a mobile vehicle platform," 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 279-284, doi: 10.1109/ROBOT.2010.5509880.

[11] Sergio Alberto Rodriguez Florez, Vincent Fremont, Philippe Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI 2008, Aug 2008, South Korea. pp.214-219, 10.1109/MFI.2008.4648067. hal 0044112

[12] Gong, X.; Lin, Y.; Liu, J. 3D LIDAR-Camera Extrinsic Calibration Using an Arbitrary Trihedron. Sensors 2013, 13, 1902–1918

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Novosad Matej** Personal ID number: **487012**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Lidar Pose Calibration Using Coded Reflectance Targets

Bachelor's thesis title in Czech:

Kalibrace polohy lidaru pomocí kódovaných odrazných značek

Guidelines:

Study the state of the art of using reflectance information of lidar responses for detection of objects in driving scenarios (e.g., road markings).
Propose and build planar targets (markers) with an identity encoded in a reflectance allowing extraction of position and code from 3D data with reflectance provided by a lidar.
Propose and implement the detection of the target's position and code.
Verify the possibility of simultaneous detection of the targets by both camera and lidar.
Use the targets for calibration of a relative pose of two lidars or lidar and camera.
Verify the system on a real-data from the multi-camera multi-lidar platform provided by our laboratory.

Bibliography / sources:

[1] Hartley, Richard and Zisserman, Andrew – Multiple View Geometry in computer vision – Cambridge university press, 2003.
[2] Farouk Ghallabi, Fawzi Nashashibi, Ghayath El-Haj-Shhade, Marie-Anne Mittet – LIDAR-Based LaneMarking Detection For Vehicle Positioning in an HD Map – 21th IEEE Intl. Conf. on Intelligent Transportation Systems, Nov 2018, Maui, Hawaii United States.
[3] Hata, A. Wolf, D. –Road marking detection using LIDAR reflective intensity data and its application to vehicle localization – 17th IEEE Intl. Conf. on Intelligent Transportation Systems, Qingdao, 2014.
[4] Zhou, L. and Li, Z. and Kaess, M. – Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences – In Proc. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, Oct. 2018.

Name and workplace of bachelor's thesis supervisor:

Ing. Martin Matoušek, Ph.D., Robotic Perception, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **29.01.2021** Deadline for bachelor thesis submission: **13.08.2021**

Assignment valid until: **30.09.2022**

Ing. Martin Matoušek, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature